



PROYECTO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

APLICACIÓN MÓVIL PARA LA COMUNICACIÓN INTERNA DE UNA EMPRESA

Autor: Diego Muñoz Herrero

Tutores: Rafael Sotomayor Fernández

Luis Miguel Sánchez García

Colmenarejo, 22 Junio de 2014



Índice de contenido

Índice de contenido.....	2
Índice de Ilustraciones	5
Índice de Tablas.....	6
1. INTRODUCCIÓN.....	8
1.1 Motivación	8
1.2 Objetivos	9
1.3 Visión.....	10
1.4 Estructura del documento.....	10
1.5 Glosario de términos y acrónimos	12
2. ESTADO ACTUAL DEL MERCADO	13
2.1 Sistemas operativos móviles	13
2.1.1 iOS	14
2.1.1.1 Historia	14
2.1.1.2 Características	15
2.1.1.3 Arquitectura	16
2.1.2 Android.....	17
2.1.2.1 Historia	17
2.1.2.2 Arquitectura	20
2.1.2.3 Intent	21
2.1.2.4 Actividades	21
2.1.2.5 Services	23
2.1.3 Windows Phone	25
2.1.3.1 Historia	25
2.1.3.2 Características	26
2.1.4 RIM (BlackBerry).....	28
3. SERVICIOS DE INCIDENCIAS WEB	29
3.1 Hidra	29
3.2 Mantis.....	31
3.3 Horde.....	34
4. PLATAFORM AS A SERVICE	35
4.1 Google App Engine	36
4.2 Heroku	37



4.3	Jelastic	38
5.	ANÁLISIS DE LA APLICACIÓN.	39
5.1	Detalle de la aplicación	39
5.2	Requisitos software.....	40
6.	DISEÑO	49
6.1	Lenguaje de programación.....	49
6.1.1	Aplicación móvil	49
6.1.2	Servidor	50
6.2	Base de datos	50
6.2.1	Aplicación móvil	50
6.2.2	Servidor	52
6.3	Comunicaciones	54
6.4	Interfaz de usuario	58
6.4.1	Manejo de incidencias Activity.....	58
6.4.2	Validar Usuario Activity	59
6.4.3	Manejo de incidencias Activity.....	60
6.4.4	Menu	60
6.4.5	Menu2	60
6.4.6	Registrar Usuario.....	60
6.4.7	Generar estadísticas de compresión y uso.	61
6.4.8	Acciones globales de aplicación.	61
7.	DESARROLLO DEL CLIENTE	62
7.1	Aplicación móvil	62
7.1.1	AsyncTask	63
7.1.2	ValidarUsuario.....	64
7.1.3	Menu	65
7.1.4	Menu2	67
7.1.5	Compresión de incidencias.....	68
7.1.6	Acciones Globales.....	69
7.1.7	Main Activity	70
7.1.8	Objeto Global	70
7.1.9	BDHelper	71
7.1.10	AndroidManifest	72
7.1.11	XML.....	73



7.1.12	Bibliotecas	74
8.	DESARROLLO DEL SERVIDOR	75
8.1	Servidor	75
8.1.1	InciController.....	75
8.1.2	Operaciones	77
8.1.3	Compression.....	78
8.1.4	DataStore.....	79
8.1.5	Interfaces.....	80
8.1.6	Bibliotecas	80
9.	TIMELINE	81
9.1	Análisis.....	83
10.	PRUEBAS DE EVALUACIÓN	84
10.1	Pruebas Unitarias	85
10.1.1	ValidarUsuario.....	86
10.1.2	Menu	87
10.1.3	Menu2	88
10.1.4	Mensajes Incidencias	89
10.2	Pruebas de aceptación (Cliente)	90
10.3	Conclusiones de las pruebas de evaluación	91
11.	PRESUPUESTO	92
12.	INSTALACIÓN Y CONFIGURACIÓN	95
12.1	Manual de instalación del servidor	95
12.2	Manual de instalación del cliente	99
12.3	Manual de uso del cliente	100
13.	TRABAJOS FUTUROS.....	103
14.	AGRADECIMIENTOS.....	104
15.	BIBLIOGRAFÍA.....	105
	Bibliografía	105



Índice de Ilustraciones

Ilustración 1 - OS Mobile Marketshare.	14
Ilustración 2 - IOS7 iPhone 5s.....	15
Ilustración 3 - IOS7	16
Ilustración 4 - Arquitectura IOS.....	17
Ilustración 5 - Android.....	17
Ilustración 6 - Evolución SO en Android.....	19
Ilustración 7 - Arquitectura Android	20
Ilustración 8 - Intent.....	21
Ilustración 9 - Ciclos de actividad	22
Ilustración 10 - Services.....	23
Ilustración 11 - Content Provider	24
Ilustración 12 - Windows Phone	25
Ilustración 13 - Arquitectura Windows	26
Ilustración 14 - Windows Phone – Interfaz Metro	27
Ilustración 15 - Blackberry Logo	28
Ilustración 16 - Hidra	29
Ilustración 17 - Ventana principal Hidra.....	30
Ilustración 18 - Generación incidencia Hidra	31
Ilustración 19 - Mantis.....	32
Ilustración 20 - Incidencias Mantis.....	33
Ilustración 21 - GAE	36
Ilustración 22 - Heroku.....	37
Ilustración 23 - Jelastic	38
Ilustración 24 - Server Database	53
Ilustración 25 - Server PaaS.....	54
Ilustración 26 - Diagrama secuencia	56
Ilustración 27 – Componentes aplicación	57
Ilustración 30 - Desviación	83
Ilustración 31 - Descarga Eclipse	95
Ilustración 32 - Añadir repositorio	96
Ilustración 33 - Añadir paquetes	97
Ilustración 34 - Crear aplicación GAE	97
Ilustración 35 - Deploy hacia GAE	98
Ilustración 36 - Manager SDK	99
Ilustración 37 - Android SDK Manager	99
Ilustración 38 - Pantalla Inicio	100
Ilustración 39 - Menu 1	101
Ilustración 40 - Menu2	101
Ilustración 41 - Generar incidencia.	102



Índice de Tablas

Tabla 1 - Glosario de términos y acrónimos.....	12
Tabla 2 - Glosario de términos y acrónimos 2.....	12
Tabla 3 - RF-01.....	40
Tabla 4 - RU-01.....	41
Tabla 5 - RF-02.....	41
Tabla 6 - RF-03.....	41
Tabla 7 - RF-04.....	42
Tabla 8 - RF-05.....	42
Tabla 9 - RF-06.....	42
Tabla 10 - RF-07.....	43
Tabla 11 - RF-08.....	43
Tabla 12 - RF-09.....	43
Tabla 13 - RF-10.....	44
Tabla 14 - RU-02.....	44
Tabla 15 - RU-03.....	44
Tabla 16 - RU-04.....	45
Tabla 17 - RU-05.....	45
Tabla 18 - RU-06.....	45
Tabla 19 - RF-11.....	46
Tabla 20 - RF-12.....	46
Tabla 21 - RF-13.....	46
Tabla 22 - RF-14.....	47
Tabla 23 - RF-15.....	47
Tabla 24 - RU-07.....	47
Tabla 25 - RU-08.....	48
Tabla 26 - Identificador / Mensaje.....	76
Tabla 27 - Tareas Ideal.....	81
Tabla 28 - Tareas Real.....	82
Tabla 29 - Hardware de prueba.....	84
Tabla 30 - Prueba Unitaria 1.....	86
Tabla 31 - Prueba Unitaria 2.....	86
Tabla 32 - Prueba Unitaria 3.....	86
Tabla 33 - Prueba Unitaria 4.....	87
Tabla 34 - Prueba Unitaria 5.....	87
Tabla 35 - Prueba Unitaria 6.....	87
Tabla 36 - Prueba Unitaria 7.....	87
Tabla 37 - Prueba Unitaria 8.....	88
Tabla 38 - Prueba Unitaria 9.....	88
Tabla 39 - Prueba Unitaria 10.....	88
Tabla 40 - Prueba Unitaria 11.....	89
Tabla 41 - Prueba Unitaria 12.....	89
Tabla 42 - Prueba Unitaria 13.....	89



Tabla 43 - Pruebas de aceptación	90
Tabla 44 - Presupuesto 1.....	92
Tabla 45 - Presupuesto 2.....	92
Tabla 46 - Presupuesto 3.....	93
Tabla 47 - Presupuesto 4.....	93
Tabla 48 - Presupuesto 5.....	93
Tabla 49 - Gastos fungibles	94
Tabla 50 - Coste total proyecto.....	94
Tabla 51 - Roles	100



1.INTRODUCCIÓN

1.1 Motivación

Los smartphones (móviles con avanzadas características), que actualmente disponen de: conexión a Internet, capacidades multimedia y alta conectividad. Han tenido un crecimiento exponencial a lo largo de estos años. Se prevé que en año 2015 el acceso a la red se realizará mayoritariamente por estos dispositivos, que desde sistemas de escritorio (ordenadores, principalmente).

En comparación con hace 10 años, los smartphones son más portables y poseen mayores características hardware que antaño (mayor CPU, RAM, conectividad). El desarrollo se ha convertido en un factor clave que determina el éxito comercial de una determinada plataforma. Esto ha llevado a los principales proveedores de sistemas operativos móviles a proporcionar herramientas de desarrollo (APIs, emulación completa en entornos) para construir aplicaciones, incluso si la plataforma no es completamente abierta.

El desarrollo de aplicaciones en Android ha crecido de manera exponencial a lo largo de estos años, como principal baza nos encontramos la libertad de distribución de aplicaciones (puede ser a través de Google Play o de manera externa) y la cantidad elevada de manuales y tutoriales para desarrollar aplicaciones.

El buen funcionamiento de un centro educativo, depende directamente de ciertos factores (entre ellos el buen funcionamiento de sus equipos informáticos y telemáticos). Todas las herramientas que nos ayuden a mantener un mejor control en la detección y reparación de incidencias en los equipos informáticos, influirá directamente en el buen funcionamiento del centro educativo.

Actualmente la universidad dispone de la plataforma Hidra, para detectar y poder hacer frente a las incidencias informáticas que se encuentran en nuestra universidad. El principal problema que encontramos, es la necesidad de disponer de un ordenador para crear / visualizar, las incidencias. Es decir, un alumno que detecte una incidencia ha de comunicárselo a un becario, y este crea una nueva entrada (teniendo que ir hasta un puesto informático, abriendo la interfaz y generando la incidencia), para que pueda verlo el becario oportuno. También puede realizarse, con el navegador desde el dispositivo móvil, pero es muy engorroso dado el tamaño de letra y la necesidad de una buena fluidez en la conexión.

El presupuesto actual de la universidad no inhiere en gastos para desarrollar e implementar un nuevo producto, centrarse en el desarrollo y las pruebas.



Este trabajo fin de grado toma las oportunidades antes mencionadas, y las introduce dentro de una aplicación móvil para el control de incidencias.

1.2 Objetivos

La finalidad de este proyecto es el estudio y desarrollo de una aplicación para el sistema operativo Android que permita el manejo de incidencias dentro de un ámbito determinado y en tiempo real. Para esta aplicación tomaremos como punto de partida las incidencias informáticas que pueden producirse en un aula informática.

El manejo de incidencias en tiempo real para dispositivos móviles será la línea de actuación principal para este trabajo, teniendo los siguientes requisitos mínimos:

- Selección de incidencias según requerimientos de la propia incidencia: Error de acceso al sistema, de periféricos (ratón / teclado), error en software y otros.
- Agrupación de las incidencias según identificador en la interfaz del becario.
- Comprobación del estado de la incidencia en tiempo real, estado de su resolución.
- (FUTURO) agrupación de incidencias y estudio por conjuntos, para dar como resultado gráficos de estudio.
- (FUTURO) Recepción de incidencias cercanas basadas en la localización actual del becario, para su rápida resolución.

Todo esto con una interfaz clara y sencilla, para dar mayores facilidades de comprensión y resolución entre los usuarios.

Suponemos un balance parejo de los alumnos que se matriculan en la universidad y los que finalizan sus estudios, en líneas futuras (suponiendo un crecimiento exponencial de los alumnos) trabajaremos sobre los parámetros de conexión entre el servidor y la aplicación para reducir los tiempos de carga y descarga.

De momento, suponemos un manejo fluido de la información y no conexiones masivas al servidor que sean capaces de ralentizar nuestra aplicación hasta niveles no funcionales, aun así comentaremos en este documento posibles mejoras y sus implementaciones para dar solución a estos requerimientos.

1.3 Visión

La aplicación constará de dos interfaces móviles diferenciadas: una enfocada a la creación de incidencias para el usuario y otra determinada para la recepción de incidencias con su correspondiente resolución, por parte de un operario (en nuestro caso, un becario). Esta funcionalidad se cribará al inicio de la aplicación.

Por otro lado, desarrollaremos un servidor que sea capaz de almacenar las incidencias en forma de base de datos, con capacidad de n accesos de lectura por parte de los becarios y n accesos de escritura por parte de los alumnos.

Para lograr el objetivo se debe además alcanzar unos objetivos secundarios:

- Conocer los principios del lenguaje de programación Java para Android.
- Diseñar y desarrollar una aplicación móvil completa.
- Diseñar e implementar la comunicación entre los subsistemas.

1.4 Estructura del documento

. **Capítulo 1, Introducción:** Mostraremos una breve descripción de los objetivos y la visión que acometeremos para la realización de este proyecto fin de grado, explicado paso por paso en plazos, tiempos y tareas.

. **Capítulos 2 / 3 / 4, Estado actual del mercado:** Análisis del mercado y comparativa de sistemas operativos para dispositivos móviles, así como los distintos servicios de incidencias web y la variedad de plataformas web que pueden servirnos para dar soporte, en forma de servidor, a nuestra aplicación. Las opciones más plausibles, serán analizadas para dar con la más eficiente y acorde a nuestros requerimientos, con la finalidad de obtener una solución elegida en base a un intensivo estudio.

. **Capítulo 5, Análisis:** Detalles exhaustivos de la aplicación, así como de los requisitos hardware – software para su óptima eficiencia. Se analizarán los casos de uso y los requisitos software.

. **Capítulo 6, Diseño:** En este capítulo se procede a diseñar el sistema utilizando como los capítulos anteriores. Aquí se diseñarán las estructuras de datos, las comunicaciones y servicios necesarios junto con la interfaz de usuario.



- . **Capítulos 7, Desarrollo Cliente:** Implementación y análisis detallado de la aplicación, perseguimos como objetivo que la aplicación sea accesible y mejorable, en un futuro, por terceras personas. De esta manera, cualquier desarrollador puede utilizar este apartado como guía. Se desarrollará el cliente y la aplicación con su interfaz.
- . **Capítulo 8, Desarrollo Servidor:** Implementación y análisis detallado de la aplicación, perseguimos como objetivo que la aplicación sea accesible y mejorable, en un futuro, por terceras personas. De esta manera, cualquier desarrollador puede utilizar este apartado como guía. Se desarrollará el servidor.
- . **Capítulo 9, Timeline:** Consecución del proyecto en plazos, tiempos y tareas.
- . **Capítulo 10, Pruebas de evaluación:** Estudio de las pruebas realizadas a la aplicación, tanto de cliente como unitarias.
- . **Capítulo 11, Presupuesto:** Analizaremos los costes inherentes al proyecto, con un estudio de pay-back
- . **Capítulo 12, Instalación y configuración:** Puesta a punto de nuestro proyecto.
- . **Capítulo 13, Trabajos futuros:** ¿Cómo podemos mejorar la aplicación?
- . **Capítulo 14, Agradecimientos**
- . **Capítulo 15, Bibliografía**

1.5 Glosario de términos y acrónimos

Tabla 1 - Glosario de términos y acrónimos

Término	Descripción
Incidencia	Alteración del funcionamiento normal en un componente, sistema o programa que requiera de la intervención de un profesional para su resolución.
Sistema Gestor de Incidencias	Encargado de gestionar las incidencias que aparecen en el sistema.
Identificador Incidencia	Único para cada incidencia, determinado por el tipo y nombre.
Urgencia de la incidencia	Brevedad con la que ha de ser resuelta la incidencia encontrada. A mayor nivel de urgencia, mayor necesidad de una rápida actuación
Tipo de la incidencia	Criterio que sirve para cribar las incidencias por el tipo de suceso.

Tabla 2 - Glosario de términos y acrónimos 2

Término	Descripción
Software Development Kit	Software Development Kit
Mbps	Mega bits por segundo
UMTS	Universal Mobile Telecommunications System.
3G	Tercera generación de transmisión de voz y datos a través de telefonía móvil mediante UMTS.
Android	SSOO basado en Linux, perteneciente a Google enfocado en dispositivos móviles y tablets.
iOS	iPhone Operating System, SSOO de Apple para dispositivos móviles.
iPhone	Smartphone de Apple.
iPad	Tablet de Apple.
iPod	Reproductor multimedia de Apple.
Tablet	Tipo de computadora portátil, de mayor tamaño que un teléfono inteligente o una PDA, integrado en una pantalla táctil (sencilla o multitáctil) con la que se interactúa primariamente con los dedos, sin necesidad de teclado físico ni ratón.



2. ESTADO ACTUAL DEL MERCADO

En esta sección aportaremos una visión global de las tecnologías usadas para la consecución de este proyecto fin de grado. También se aportarán posibles alternativas a utilizar.

La valoración que realizaremos, tendrá en cuenta las principales tecnologías y opciones, con las que podríamos realizar nuestro proyecto.

Como resultado final, obtendremos una elección tecnológica clara y concisa.

2.1 Sistemas operativos móviles

En esta sección se expondrán la historia y características de los sistemas operativos más destacados para teléfonos móviles inteligentes (smartphones).

Los sistemas operativos que se analizarán serán:

- iOS
- Android
- Windows Phone
- Blackberry OS (RIM)

El mercado de los dispositivos móviles inteligentes (smartphones) ha visto un notable incremento a lo largo de estos últimos años, como podemos observar en la siguiente [Ilustración 1](#) - OS Mobile Marketshare., existen distintos distribuidores (con sus correspondientes dispositivos) que dominan dicho mercado.

El crecimiento de Android, ha sido muy notable a lo largo de este periodo, seguido muy de cerca por iOS.

La capacidad de instalarse en múltiples dispositivos, junto al entorno de desarrollo sobre aplicaciones que provee Android, han representado dos grandes bazas para este gran crecimiento.

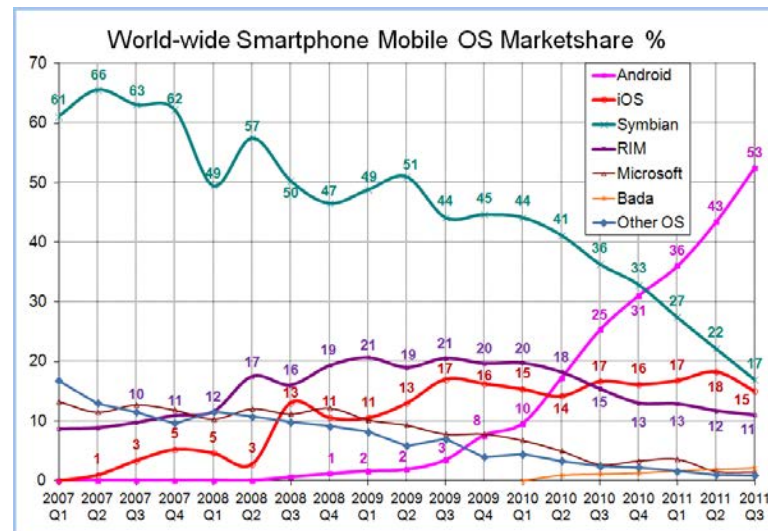


Ilustración 1 - OS Mobile Marketshare.

2.1.1 iOS

El sistema operativo desarrollado por Apple Inc. Originalmente fue desarrollado para el iPhone, para después ser usado en el resto de dispositivos Apple.

2.1.1.1 Historia

La primera versión de iOS fue presentada en 2007 junto con el primer iPhone. Versiones posteriores de iOS acompañaron a cada versión de iPhone. Actualmente, se presenta el sistema operativo alrededor de Junio y el dispositivo nuevo que lo integrará en Septiembre.

La independencia de sistema operativo, se produjo a raíz de la salida al mercado del iPhone 4 (en el año 2010), siendo bautizado como iOS.

Sucesivas versiones se han actualizado con el paso de los años, a través de “Keynote”, un evento mundialmente seguido desde todos los rincones del mundo. EL último dispositivo presentado ha sido el [Ilustración 2](#) - IOS7 iPhone 5s , aunque ya hay rumores que apuntan a un iPhone 6 con dos tamaños de pantalla. El último iOS presentado ha sido el 8, en este mismo mes.



Ilustración 2 - IOS7 iPhone 5s

2.1.1.2 Características

Las aplicaciones del sistema operativo iOS están escritas en Objective-C y se comunican con el hardware a través de un conjunto de APIs ya publicados. Ofrece distintas capas de abstracción para crear menús en pantalla, e integrarlos con tecnología 2D y 3D, servicios de localización y conectividad a la red.

iOS logra un aislamiento dentro de la aplicación a través de un “Sandbox” (caja de arena) similar al utilizado en Mac OS X, en el que un archivo de políticas restringe el acceso a ciertas características del dispositivo y los datos contenidos en él.

Por defecto, la aplicación puede leer o escribir datos fuera de su propio directorio, los cuales pueden incluir archivos del sistema, recursos compartidos y el propio kernel (Ilustración 4 - Arquitectura IOS).

Los desarrolladores que deseen publicar aplicaciones para el sistema operativo iOS han de obtener antes la aprobación por parte de Apple

Apple no ha publicado información detallada adecuando los criterios de aceptación de aplicaciones desarrolladas, la creencia general que poseen los desarrolladores es que Apple emplea bots para el análisis de aplicaciones y manuales de verificación. Estos bots, tendrían una serie de algoritmos o directrices enteras que determinarían si una aplicación es válida o no.



Ilustración 3 - IOS7

Si una aplicación se clasifica como apta para su distribución pública, Apple lo firma digitalmente y lo libera al centro de intercambio de software Apple, el iTunes App Store.

Apple rechaza aplicaciones que violan las leyes de protección intelectual o los términos de desarrolladores de servicio.

2.1.1.3 *Arquitectura*

En un principio iOS solo podía ejecutar una aplicación a la vez, la multitarea estaba reservada a aplicaciones del sistema, no fue hasta iOS 4 cuando Apple introdujo la multitarea en el API que liberó a los desarrolladores.

Todos los dispositivos iOS cuentan con cámara, WIFI y GPS. Algunos integran conexión 3G (iPhone y algunas versiones de iPad) para disponer de este tipo de conexión en cualquier lugar.

Por último, la Ilustración [4](#) - Arquitectura IOS se compone de 3 capas. En el primer nivel encontramos la capa de aplicaciones, conocida como Cocoa Touch, una API del sistema operativo que proporciona una capa de abstracción para crear aplicaciones. Después encontramos la capa Core Service donde se encuentran las API de seguridad del sistema basadas en los servicios de la última capa (llamada Core OS) es la capa del núcleo del sistema.



Ilustración 4 - Arquitectura IOS

2.1.2 Android



Ilustración 5 - Android

Sistema móvil empleado en la realización de este proyecto.

2.1.2.1 Historia

Android es un sistema operativo multiplataforma de código abierto, respaldado por Google y basado en Linux. Android alimenta una gran variedad de teléfonos inteligentes, tablets y netbooks de muchos fabricantes. Desde su aparición y primer lanzamiento en 2008, la base de desarrollo Android ha sido testigo de un rápido crecimiento y desarrollo.

Fundada como Android Inc. para ser comprada en 2005 por Google, Android siguió siendo desarrollado por la Open Handset Alliance, unión de fabricantes de software y hardware como T-Mobile, HTC o Qualcomm, liderado por Google.



El SDK de Android se presentó en 2007, pero no fue hasta un año después cuando vio la luz el primer Smartphone con este sistema operativo, un HTC, con Android 1.0 que incluía los servicios básicos de Google.

Desde la primera [versión Ilustración 6](#) - Evolución SO en Android hasta la actual 4.22, Android ha impuesto grandes cambios tanto estéticos como internos.

En 2009 se actualizó Android a 1.5 Esta versión mejoraría la velocidad general del sistema, la localización GPS y mayor integración con servicios como YouTube.

Más tarde en el mismo año saldría la versión 1.6 Donut, que incluiría la búsqueda escrita y por voz, un indicador de uso de batería y daría soporte para CDMA.

Un mes después, se lanzó otra nueva versión, la 2.0 (llamada Eclair) corrigiendo fallos de la versión anterior. En esta versión se añadirían sincronización de contactos, soporte para Bluetooth 2.1, una nueva interfaz para el navegador con soporte para HTML 5.

En 2010 se liberaron Android 2.1 y 2.2 Froyo, que incluiría Adobe Flash 10.1, teclado en varios idiomas y soporte para punto de acceso (APN).

A finales de 2010 salió la hasta ahora versión más popular de Android, la 2.3 Gingerbread. Esta versión incorporaría llamadas a través de internet, NFC, un nuevo teclado y nuevas funcionalidades (como el copiado y pegado de texto).

La siguiente revolución de Android fue únicamente en el soporte para tablets, esta versión se llama HoneyCromb 3.0.

En octubre 2011 Google presentó la renovación más importante de Google en la que Android alcanzaría la madurez como Sistema Operativo. Esta versión unificaría la versión de tablets y móviles con interfaz optimizada.

En el Google I/O de 2012, en junio, se presentó la versión 4.1 Jelly Bean. Esta nueva versión sería pionera en el primer Tablet de Google, el Nexus 7, fabricado por Asus. Jelly Bean mejoraría la fluidez general del sistema con el conocido como Project Butter. Además mejoraría el rendimiento y la compatibilidad de HTML 5 con el navegador.

La última versión liberada se considera una actualización de Jelly Bean, nombrándose como 4.3

La fragmentación entre dispositivos es uno de los mayores problemas de Android, ya que existen muchos dispositivos en el mercado con muchas versiones instaladas diferentes.



Ilustración 6 - Evolución SO en Android

La última versión conocida es 4.4 KitKat, se espera el lanzamiento de Android 5.0 para este año 2014.

2.1.2.2 Arquitectura

La arquitectura de Android, consta de cinco componentes como podemos observar en la siguiente [Ilustración 7 - Arquitectura Android](#)

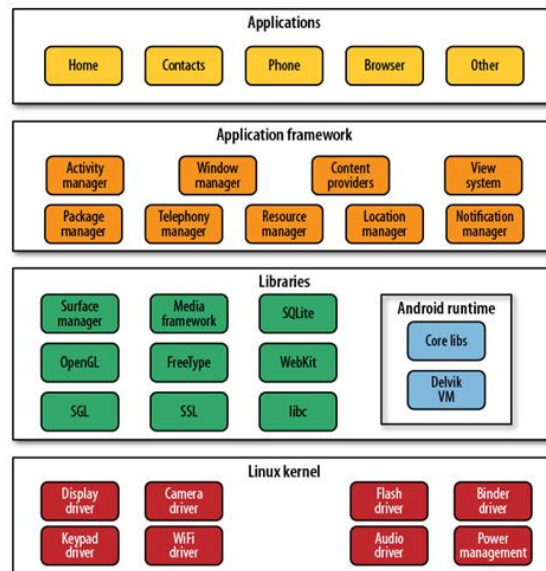


Ilustración 7 - Arquitectura Android

Applications: La última capa, en la arquitectura Android. Algunas aplicaciones vienen incluidas en el sistema operativo para poder interactuar con el mismo, por ejemplo: los mapas, lista de tareas, el calendario, etc.

Application framework: Es la capa disponible para crear las aplicaciones, por parte de los desarrolladores. Éstos tienen acceso total a la API, con una fuerte arquitectura basada en la reutilización de componentes.

Libraries: Bibliotecas incluidas de C / C++, que son utilizadas por diversas partes del sistema operativo. Son ofrecidas a los desarrolladores a través de “Application Framework”

Linux Kernel: Al estar basado en el kernel de Linux, Android comparte la gestión de procesos y servicios básicos con éste. El kernel actúa como una capa entre el hardware y el resto del software.

2.1.2.3 *Intent*

Un intent, es una descripción abstracta de una operación a realizar. En nuestra aplicación, lo emplearemos para pasar de una actividad (función de un determinado menú, por ejemplo hacer click en un botón) hacia otra actividad necesaria (la consecución de ese botón.)

Un intent provee facilidad de unión entre dos fragmentos de código, en diferentes partes de la aplicación.

En este [Ilustración 8 - Intent](#), declararemos la actividad que nos encontramos y nuestra actividad de destino:

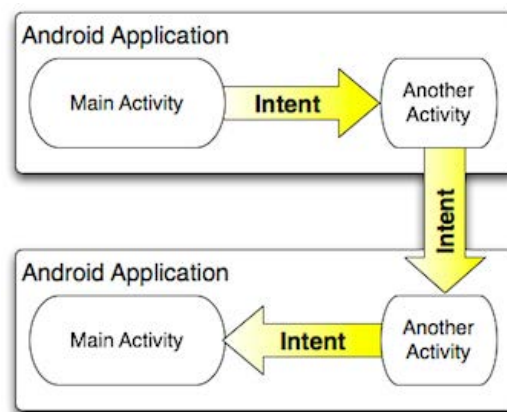


Ilustración 8 - Intent

2.1.2.4 *Actividades*

Una aplicación Android, está formada por un conjunto de elementos básicos de interacción con el usuario, conocidos como actividades. Además de actividades, una aplicación puede contener servicios.

El “Activity Manager” se encarga de manejar las actividades (creándolas, destruyéndolas, reanudándolas o parándolas). Cuando un usuario lanza una aplicación por primera vez, el Activity Manager trae la actividad a primer plano. Cuando el usuario cambie a otra pantalla, el Activity Manager guardará ese activity en memoria, para cuando el usuario vuelva al primer Activity tardará mucho menos en lanzarlo. Los activities que no son usados en cierto tiempo, son destruidos automáticamente para dejar hueco a nuevos activities.

En la siguiente [Ilustración 9](#) - Ciclos de actividad, podemos observar el ciclo de una actividad y las acciones para pasar de una a otra:

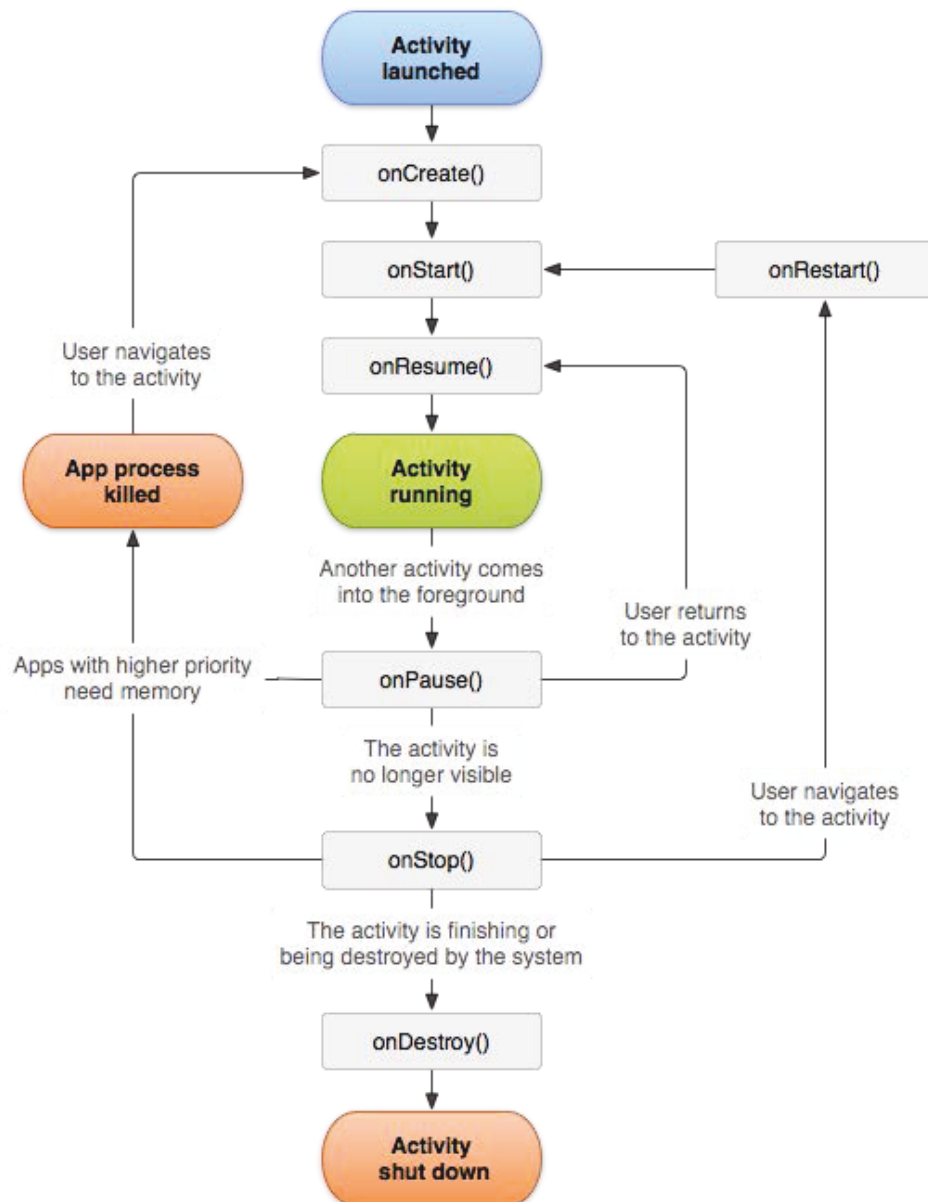


Ilustración 9 - Ciclos de actividad

Android es sensible al ciclo de vida de una actividad, por lo tanto es necesario comprender y manejar los eventos relacionados con el ciclo de vida:

- **Running:** La actividad es visible y tiene los recursos totalmente asignados para enfocarse en su función.
- **Paused:** La actividad es visible, pero no tiene los recursos focalizados.
- **Stopped:** Una actividad no es visible.
- **Destroyed:** La actividad termina al invocarse el método *finish()*.

2.1.2.5 Services

Otro gran componente de Android, son los Services. Se ejecutan en segundo plano, sin ningún tipo de interfaz. Son útiles para acciones que tardan en ejecutarse y necesitan independencia de lo que ocurre en la interfaz.

El ciclo de vida de un Service puede verse en la siguiente [Ilustración 10 - Services](#).

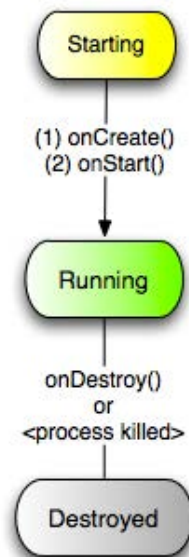


Ilustración 10 - Services

Cuando queremos pasar mucha información o manejar muchos datos constantemente se deben usar los Content Providers, en vez de los intents.

Los content providers, son usados constantemente en las aplicaciones de Android, normalmente para poder ofrecer una gran cantidad de datos (contactos, incidencias...) para cualquier aplicación que los solicite. En la siguiente [Ilustración 11 - Content Provider](#), podemos observar la estructura de un content provider.

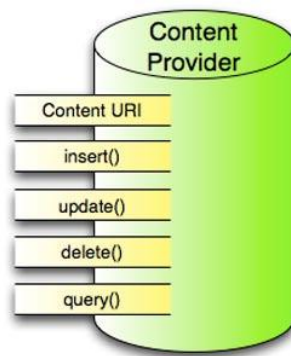


Ilustración 11 - Content Provider

El resto de componentes, forman una aplicación en conjunto dentro del “application context”

Recoge los procesos y variables que se ejecutan en la aplicación, permitiendo reunir toda la información y recursos para ser compartidos entre el resto de componentes. Permite que desde cualquier punto de la aplicación, llamar a la función “getApplicationContext()” para recuperar todos los datos asociados a esta. Es usado constantemente a lo largo del proyecto

2.1.3 Windows Phone



Ilustración 12 - Windows Phone

Windows Phone es un sistema operativo para móviles desarrollado por Microsoft, como sucesor de la plataforma Windows Mobile. Está enfocado en un mercado para el consumo generalista.

2.1.3.1 *Historia*

Windows Phone tiene como raíz el anterior Windows Mobile.

Este fue presentado en el año 2000 y en los posteriores años fue integrándose con los servicios de Microsoft como el MSN Messenger, el Media Player, Internet Explorer u Office Mobile. No fue conocido como Windows Mobile hasta el año 2003 e introdujo soporte Multitouch ya en el año 2009.

Windows Mobile fue presentado en el año 2000 y posteriormente fue integrándose con los servicios de Microsoft como Internet Explorer. En el año 2003, fue conocido como Windows Mobile.

Pero la verdadera apuesta por hacerse un hueco en un mercado liderado por Android y iOS fue en 2010 cuando Microsoft presentó el Windows Phone 7, que posteriormente en el año 2012 evolucionaría a la versión 8 para alcanzar una madurez y una calidad a la altura de sus competidores en el sector.

2.1.3.2 Características

El núcleo del sistema operativo de Windows Phone está basado en el Windows Embedded CE, para desarrollar las aplicaciones podemos utilizar Silverlight o XNA Framework.

Silverlight permite desarrollar aplicaciones variadas, con una gran base en efectos visuales y transiciones entre ventanas. Microsoft .NET Compact Framework se encuentra dentro de Silverlight.

Los desarrolladores pueden descargar herramientas gratuitas junto al SDK, para desarrollar aplicaciones. Microsoft permite publicar hasta 100 aplicaciones de forma gratuita

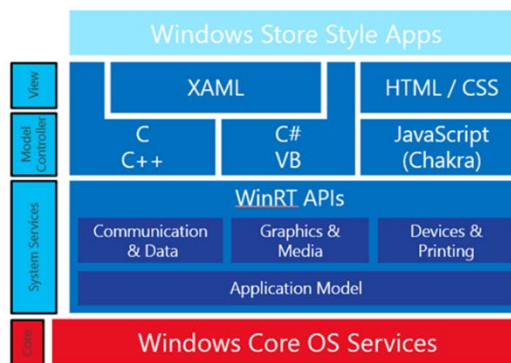


Ilustración 13 - Arquitectura Windows

La **arquitectura** que podemos observar en la siguiente [Ilustración 13 - Arquitectura Windows](#), consta de un framework para desarrollar las apps del sistema y controladores para los distintos componentes hardware que pueda tener el dispositivo, enfocado a una mayor diversidad de fabricantes.

Windows Phone, trata de diferenciarse en su navegación a través de una pantalla de inicio compuesta de mosaicos llamados “Live Tiles”, dichos mosaicos (como observamos en la [Ilustración 14 - Windows Phone – Interfaz Metro](#)) están dispuestos de manera dinámica y enlazan con el resto de aplicaciones y funciones del teléfono o tablet. Esta característica fue portada a Windows 8 a través de su interfaz “metro”, muy criticada por parecer un formato Tablet adaptado a un sistema de escritorio



Ilustración 14 - Windows Phone – Interfaz Metro

2.1.4 RIM (BlackBerry)



Ilustración 15 - Blackberry Logo

Research in Motion (RIM) fue el encargado de desarrollar el BlackBerry OS. Únicamente funciona en los modelos de BlackBerry e históricamente es utilizado en el sector empresarial, mediante la inclusión de características tales como push email y soporte de trabajo en grupo.

Es compatible con aplicaciones de terceros escritas en Java. Utiliza una “Sandbox” para aislar aplicaciones en tiempo de ejecución, a través de la máquina virtual Java. Los desarrolladores escribían las aplicaciones tradicionalmente en Java y a través de sitios web sin necesidad de la aprobación por parte de RIM. Esto cambió en 2009 con la aparición de BlackBerry App World, en la que los usuarios de nuevos modelos de BlackBerry pueden acceder a dicho repositorio, donde las aplicaciones sí están aprobadas por RIM a través de un dispositivo.

Los desarrolladores pueden albergar sus aplicaciones en otros servidores, a pesar de la aprobación de RIM.

BlackBerry OS ofrece a las empresas un gran filtro de control sobre los dispositivos que distribuyen a sus empleados. Los administradores pueden incluir políticas dentro de los dispositivos BlackBerry, permitiendo restringir funcionalidades. Por ejemplo pueden limitar la instalación de aplicaciones de terceros descargadas fuera del BlackBerry App World.

La **arquitectura** de Blackberry OS dispone de un sistema operativo escrito en C++ y adquiere características tales como: multitarea, comunicación entre procesos, hilos y soporte para dispositivos de entrada.

3.SERVICIOS DE INCIDENCIAS WEB

Una vez estudiados las distintas opciones en tecnología móvil que tenemos para desarrollar nuestra aplicación, será necesario evaluar los distintos servicios de incidencias web ya disponibles y en funcionamiento para entender que características hemos de portar a nuestra aplicación.

El análisis de los distintos servicios de incidencias web junto a su funcionalidad, nos ayudará a entender que características busca el usuario cuando hace uso de estas tecnologías, y por tanto esas características serán necesarias en nuestra aplicación.

Los servicios de incidencias Web son tecnologías web, que usan protocolos para almacenar y manejar incidencias sobre un determinado campo de actuación.

Permiten el manejo dinámico de una base de datos en la web, a través de una pantalla de acceso principal basada en credenciales.

Analizaremos el sistema de manejo de incidencias utilizado en la Universidad Carlos III junto a otras tecnologías como Mantis y Horde.

3.1 Hidra

Accediendo al sistema gestor de incidencias de la UC3M, primeramente nos encontraremos con esta Ilustración 16 - Hidra:



Ilustración 16 - Hidra



Los datos de acceso son comparados con los datos contenidos en la base de datos para determinar las limitaciones de rol impuestas al usuario (no tendrán las mismas capacidades un becario que el administrador del sitio).

El becario puede observar las incidencias que se han generado y actuar en consecuencia, a través de la siguiente Ilustración 17 - Ventana principal Hidra:

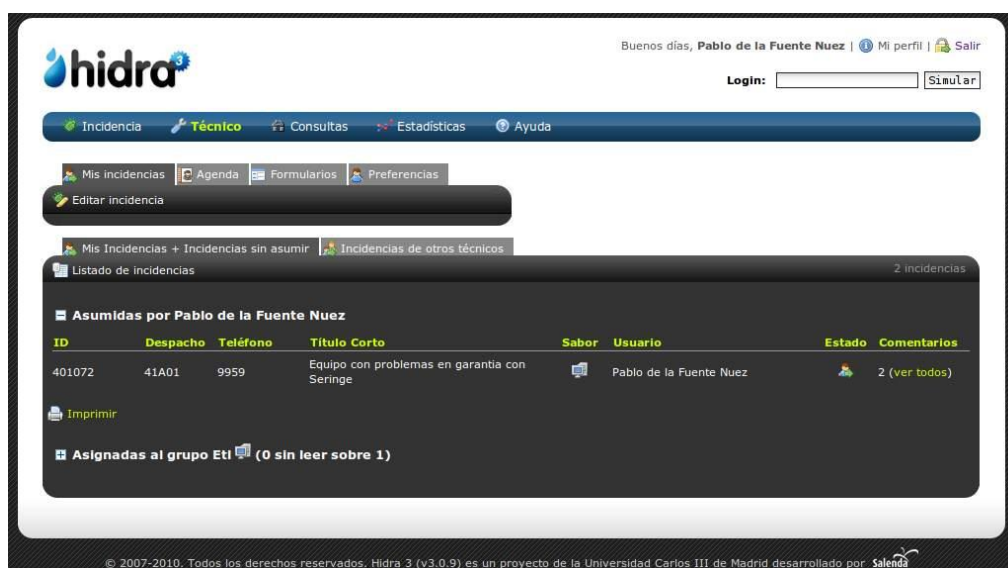


Ilustración 17 - Ventana principal Hidra

Todas las incidencias son agrupadas en esta pantalla, los identificadores (dato importante que tomaremos para la consecución de nuestro proyecto) permiten identificar unívocamente a las incidencias.

Hidra posee muchas funcionalidades buscadas en nuestra aplicación, por lo que tomaremos ideas basadas en sus manuales de usuario y experiencia de uso.

La generación de incidencias se realiza a través de la siguiente Ilustración 18 - Generación incidencia Hydra, donde podemos cumplimentar diversos campos que faciliten la organización y resolución de las mismas:



Crear incidencia

login
Buscar por: ☐ nombre ☐ despacho

Usuario: Pablo de la Fuente Nuez

Email: pfuente@pa.uc3m.es

Departamento: Gestión de Recursos

Sujeto a LOPD: ☐

Despacho: 41A01 Campus: Leganés Teléfono: 9959

Edificio: Torres Quevedo

Título corto: Equipo con problemas en garantía con Serial: Teléfono

Descripción: Hola, un equipo al arrancar se quedaba en Bios data check succesful. Pero hoy ya no ha dado señales de vida (aunque se enciende). Esta en la 70J02 y es: 111702XT0055 - jbit124
Un saludo.
Pablo.

Avisar por mail: staff@adm-it.uc3m.es

Número de incidencias: 1

Enviar Incidencia

Ilustración 18 - Generación incidencia Hydra

La incidencia se encuentra disponible, para ser consultada por los becarios, una vez que hacemos click en “Enviar incidencia” y esta aparece en la ventana principal de Hydra ([Ilustración 17 - Ventana principal Hydra](#))

3.2 Mantis

Aplicación OpenSource hecha en PHP y MsQL, constituida como solución para gestionar tareas de un equipo de trabajo. Se utiliza para testear soluciones, generar y manejar incidencias (punto que nos interesa para el devenir de nuestro proyecto) y gestionar equipos remotamente.

El gran abanico de posibilidades para su configuración, es una de las características de Mantis. Permite especificar un número indeterminado de estados para cada incidencia (abierta, encaminada, testeada...) y configurar su transición (abierta, cerrada, reabierta...)



Incluye un sistema de búsqueda muy conseguido con filtros y la introducción de diversos perfiles de administración y resolución.

Posee unos requerimientos modestos, aunque se requiere que el equipo sea capaz de ejecutar el software de un servidor.

La adaptación de Mantis hacia las nuevas tecnologías ha supuesto la portabilidad de la plataforma hacia una aplicación móvil ([Ilustración 19 - Mantis](#)), donde podemos realizar las mismas acciones que desde la aplicación web.

Esta es una de las grandes bazas de Mantis, para competir en este sector. También nos servirá para determinar qué características y funcionalidades, son las más buscadas en una aplicación móvil para la detección de incidencias.

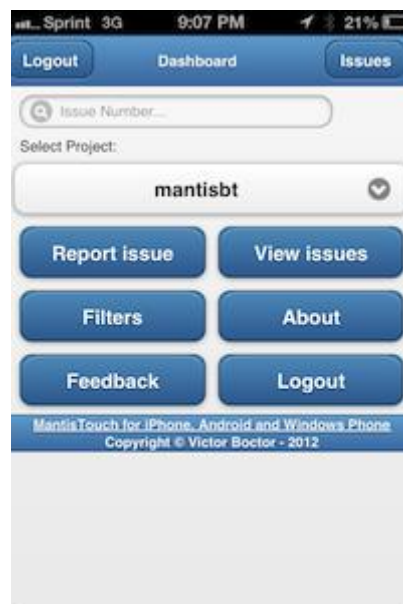


Ilustración 19 - Mantis

Las incidencias son generadas y dispuestas en la siguiente pantalla (Ilustración 20 - Incidencias Mantis), donde un becario puede acceder a ellas y disponer de los datos, para su resolución.

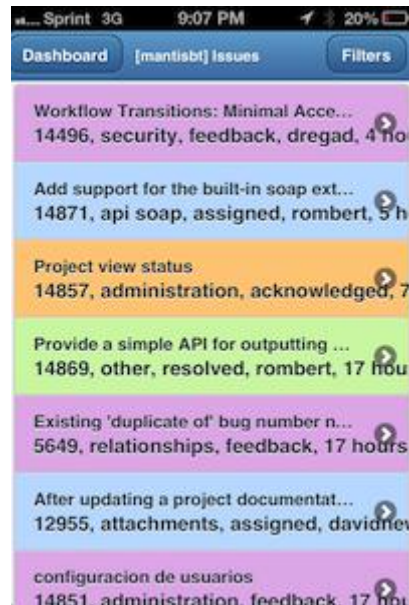


Ilustración 20 - Incidencias Mantis

La aplicación posee un carácter multiplataforma al ser soportada por Android, iOS y Windows Phone.

Esta es una característica que tendremos que evaluar en futuros desarrollos para nuestra aplicación, ya que actualmente no dispone de soporte para dispositivos fuera de Android.

La cuota de mercado, es dominada actualmente por Android seguida muy de cerca por iOS ([Ilustración 1 - OS Mobile Marketshare](#)), esto es determinante para futuras portabilidades de nuestra aplicación hacia otros sistemas operativos móviles.



3.3 Horde

Es un framework libre escrito en PHP, para el desarrollo de aplicaciones colaborativas (entre las que se encuentra la gestión de incidencias).

Horde, se compone de tres bibliotecas que proporcionan las funcionalidades básicas y que funcionan como unión entre distintas aplicaciones de usuario, gestionadas cada una por proyectos independientes.

Está basado en licencia LGPL (*Library General Public License* o Licencia Pública General para Bibliotecas de GNU)

Entre los proyectos de Horde, podemos encontrar:

- **IMP**: Sistema de correo electrónico que permite acceso a buzones POP3 / IMAP.
- **MIMP**: Derivado de IMP, con una interfaz mínima para su implementación en dispositivos móviles.
- **INGO**: Sistema de gestión y aplicación de reglas de filtrado para correo.
- **Turba**: Agenda de contactos.
- **Gollem**: Gestor de archivos.
- **Trean**: Gestor de favoritos.

La oferta de este tipo de servicios incluye a grandes empresas como Google con Google App Engine, Microsoft con Windows Azure, Red Hat con OpenShift o Amazon con Amazon Web Services entre muchos otros.

Una vez analizadas las distintas herramientas ya disponibles, que realizan la funcionalidad de manejar incidencias de manera web, tenemos una visión más clara de los requisitos que ha de tener nuestra aplicación y las funcionalidades que tenemos que implementar en ella.



4.PLATAFORM AS A SERVICE

Una vez estudiadas las distintas opciones sobre tecnología móvil a desarrollar e implementar sobre nuestra aplicación, y los distintos servicios de incidencias ya disponibles, tenemos clara la tecnología móvil en la que vamos a desarrollar la aplicación y las funcionalidades básicas que ha de tener (obtenidas del análisis de los servicios de incidencias en el punto anterior).

Ahora será necesario evaluar las distintas opciones que tenemos para implementar el servidor.

Nuestra aplicación ha de disponer de un servidor global, en el que almacenar la información de incidencias, para que sea accesible desde cualquier localización.

En esta sección expondremos las características principales de algunas plataformas de computación establecidas en “la nube”. También conocidos como Plataform As Service (PaaS)

La plataforma como servicio o platform as a service es un conjunto de servicios cloud que proporciona un entorno para el desarrollo, implantación, gestión e integración de aplicaciones en la nube. PaaS puede reducir el coste y la complejidad de su desarrollo con la utilización de herramientas y servicios basados en cloud, así como la estandarización de tareas clave pertenecientes al desarrollo de aplicaciones.

Este tipo de servicios ofrecen computación de gran escalabilidad, altas prestaciones y gran facilidad para poner en marcha una aplicación en la nube, que pueda ser empleada por diferentes personas, que lo convierte en un gran entorno a emplear en este proyecto.

Los servicios que vamos a analizar son Google App Engine, Heroku y Jelastic.

4.1 Google App Engine

Google App Engine, también conocido como “GAE”, permite crear y alojar aplicaciones web. Las aplicaciones corren en procesos aislados, también llamados sandboxed, en multitud de servidores. Ofrece procesos de desarrollo e implementación rápidos junto con una administración sencilla. Ofrece una gran facilidad a la hora de desplegar una aplicación.

Ofrece una escalabilidad automática, en sus aplicaciones web, de manera transparente. De esta manera, si la aplicación necesita mayor velocidad de procesamiento o incrementa el número de peticiones, Google App Engine aumenta los recursos de memoria y CPU.

Google App Engine soporta Java y Python, por extensión al utilizar tecnologías basadas en máquinas virtuales también soporta Scala, JRuby, Jython o PHP, entre otras

Las limitaciones de GAE se encuentran en la única posibilidad de utilizar los lenguajes arriba mencionados, además las APIs ofrecen manejar datos de una base de datos no relacional propia de Google. Para el almacenaje de los datos, GAE hace uso de “datastore”, posee una sintaxis similar a SQL, conocida como GQL en una base de datos con modelo no relacional.

GAE es de distribución gratuita para los desarrolladores con la única limitación dentro de determinados usos de CPU, memoria y almacenamiento.



Ilustración 21 - GAE

4.2 Heroku

Heroku soporta varios lenguajes como Ruby, Java, Node.js, Scala y PHP. El sistema operativo base que utiliza es Ubuntu, anteriormente Debian. Pertenece a salesforce.com



Ilustración 22 - Heroku

Es una de las primeras plataformas de computación en la nube, su objetivo principal fue soportar únicamente el lenguaje Ruby, pero posteriormente se ha extendido.

La base del sistema operativo es Debian Ubuntu.

Posee más libertad general que GAE y la gran ventaja de poder portar un servidor ya escrito en Java, para desplegarlo en Heroku, sin modificación necesaria.

Ofrece almacenamiento de datos, concretamente SQL

Su coste es gratuito hasta cierto límite de recursos, con cuotas muy parecidas a las de GAE.

4.3 Jelastic



Ilustración 23 - Jelastic

Jelastic es un proveedor de PaaS, para aplicaciones Java y PHP, ofreciendo cloud hosting. Cuenta con centros internacionales de datos y hosting. La empresa añade memoria, CPU y espacio en disco, para satisfacer las necesidades del cliente.

Jelastic no posee limitaciones o requisitos de cambio de código y ofrece un escalado automático, con gestión de ciclo de vida de aplicaciones y disponibilidad de múltiples proveedores de alojamiento en todo el mundo.

En mayo de 2013, Jelastic anunció un nuevo plug-in para la integración con NetBeans IDE. Dos semanas más tarde, la compañía lanzó Jelastic PaaS 1.9.1 que incluye las últimas versiones de los paquetes de software (incluyendo PostgreSQL 9.2.4)

El 14 de Agosto de 2013, la versión 1.9.2 fue lanzada, incluyendo FTP y FTPS, acceso a los servidores de bases de datos, capacidad para implementar PHP, proyectos de repositorio GIT con sub-módulos / dependencias y PHP versión 5.5



5. ANÁLISIS DE LA APLICACIÓN.

Una vez realizado el análisis de todas las alternativas que tenemos, en cuanto a lenguaje a implementar nuestra aplicación móvil + tecnología a implementar en nuestro servidor + características principales que ha de disponer nuestra aplicación, tenemos los conocimientos necesarios para centrarnos en el análisis exhaustivo de la aplicación para la detección de incidencias en una empresa.

5.1 Detalle de la aplicación

El objetivo de la aplicación es permitir el funcionamiento básico de un sistema gestor de incidencias en un dispositivo smartphone con sistema operativo Android, a través de una interfaz clara e intuitiva, disponiendo de un servidor GAE que permita manejar datos de manera global.

El funcionamiento básico incluirá lo siguiente:

- Posibilidad de acceder a distintas interfaces dependiendo del rol desempeñado en la Universidad (becario, alumno).
- Generar incidencias desde la aplicación, y manejarlas en el caso del becario.
- Ver el estado de las incidencias creadas y su resolución.
- Enviar feedback al desarrollador.
- Permitir inicio seguro y manejo seguro de los datos sensibles, dentro de la aplicación.
- Generar estadísticas de compresión y uso de la aplicación, para el estudio por un becario.

Por otro lado, este proyecto añadirá funcionalidades de compresión ahorrando el consumo de datos móviles, al hacer uso de las funcionalidades básicas de la aplicación. Para ello, será necesario un servidor transparente al desarrollador y usuario, que realizará la compresión de datos y cuyas funcionalidades básicas son:

- Compresión del manejo de incidencias (crear / enviar / ver el estado / descargar / resolver) para reducir el número de bytes que se transmiten por la red.
- Configuración de la aplicación para reducir más o menos el consumo de datos en función del tipo de usuario.

5.2 Requisitos software

A continuación se exponen los requisitos que detallan las características indispensables que ha de tener el sistema para funcionar correctamente.

La tabla explicativa se compone de diversos campos:

- **ID Requisito:** identificador unívoco del requisito.
- **Tipo:** funcional o de usabilidad, según corresponda de acuerdo a su cometido.
- **Evento/Caso de uso:** situación que debe ocurrir como disparador.
- **Descripción:** breve explicación del significado del requisito.
- **Justificación:** por qué es importante el requisito y qué motivación lo ha propiciado.
- **Criterio de cumplimiento:** comprobante de una implementación del requisito satisfactoria.
- **Prioridad:** relevancia del requisito.
- **Conflictos:** requisitos que no pueden ser implementados si éste requisito es implementado.

A través de estos requisitos, deduciremos las clases que construirán nuestra aplicación y el diagrama de flujo de la misma.

Los requisitos son los siguientes:

Tabla 3 - RF-01

ID Requisito	RF-01	Tipo	Funcional
Evento / Caso de uso	Registro en la aplicación.		
Descripción	El administrador a través de la ventana o interfaz de registro, da de alta al usuario.		
Justificación	Únicamente los integrantes de nuestra universidad pueden utilizar el SGI		
Criterio de cumplimiento	Abrir la ventana de registro, habilitada desde el menú de administración, seleccionar y rellenar los campos necesarios para almacenar nuevo usuario.		
Prioridad	Alta	Conflictos	#



Tabla 4 - RU-01

ID Requisito	RU-01	Tipo	Usabilidad
Evento / Caso de uso	Selección entre varios roles		
Descripción	El usuario a través del menú de preferencias selecciona entre 2 niveles de autenticación dependiendo de su rol		
Justificación	El usuario posee diversas funcionalidades en el menú dependiendo de su rol con/para la aplicación.		
Criterio de cumplimiento	Iniciar la aplicación y seleccionar la opción acorde a sus credenciales.		
Prioridad	Media	Conflictos	#

Tabla 5 - RF-02

ID Requisito	RF-02	Tipo	Funcionalidad
Evento / Caso de uso	Selección del nivel de rol: Becario.		
Descripción	Al seleccionar el rol de becario, se desactiva la opción de creación de incidencias y se habilita la posibilidad de resolverlas.		
Justificación	El becario se dedica a la resolución de incidencias, únicamente.		
Criterio de cumplimiento	Una vez seleccionada esta opción, se carga el menú perteneciente al rol Becario, donde puede manejar las incidencias y resolverlas.		
Prioridad	Alta	Conflictos	#

Tabla 6 - RF-03

ID Requisito	RF-03	Tipo	Funcionalidad
Evento / Caso de uso	Selección del nivel de rol: Becario.		
Descripción	Al seleccionar el rol de becario, se activa la posibilidad de creación de nuevos usuarios.		
Justificación	El becario únicamente será el encargado de crear nuevos usuarios en la aplicación.		
Criterio de cumplimiento	Una vez seleccionada esta opción, se carga el menú perteneciente al rol Becario, donde puede manejar esta funcionalidad.		
Prioridad	Alta	Conflictos	#



Tabla 7 - RF-04

ID Requisito	RF-04	Tipo	Funcionalidad
Evento / Caso de uso	Selección del nivel de rol: Becario.		
Descripción	Al seleccionar el rol de becario, se activa la posibilidad de generar estadísticas de compresión y uso.		
Justificación	El becario únicamente será el encargado de generar dichas estadísticas sobre el servidor.		
Criterio de cumplimiento	Una vez seleccionada esta opción, se carga el menú perteneciente al rol Becario, donde puede manejar esta funcionalidad.		
Prioridad	Alta	Conflictos	#

Tabla 8 - RF-05

ID Requisito	RF-05	Tipo	Funcionalidad
Evento / Caso de uso	Selección del nivel de rol: Becario.		
Descripción	Al seleccionar el rol de becario, se activa la posibilidad de realizar labores administrativas (suspensión del servicio), en el servidor.		
Justificación	El becario únicamente puede tener los privilegios necesarios para realizar estas tareas, con un nivel de sensibilidad alto.		
Criterio de cumplimiento	Una vez seleccionada esta opción, se carga el menú perteneciente al rol Becario, donde puede manejar esta funcionalidad.		
Prioridad	Medio	Conflictos	#

Tabla 9 - RF-06

ID Requisito	RF-06	Tipo	Funcionalidad
Evento / Caso de uso	Selección del nivel de rol: Alumno.		
Descripción	Al seleccionar el rol de Alumno, se carga la opción de creación para nuevas incidencias.		
Justificación	El estudiante genera nuevas incidencias.		
Criterio de cumplimiento	Una vez seleccionada la opción, se carga el menú perteneciente al rol Alumno, donde existe una tercera opción más que en el rol de Becario y no existe la posibilidad de resolver incidencias.		
Prioridad	Alta	Conflictos	#



Tabla 10 - RF-07

ID Requisito	RF-07	Tipo	Funcionalidad
Evento / Caso de uso	Selección del nivel de rol: Alumno.		
Descripción	Al seleccionar el rol de Alumno, se carga la opción de visualización de incidencias ya creadas por el mismo.		
Justificación	El estudiante visualiza sus incidencias creadas.		
Criterio de cumplimiento	Una vez seleccionada la opción, se carga el menú perteneciente al rol Alumno, donde existe una opción para visualizar sus incidencias.		
Prioridad	Alta	Conflictos	#

Tabla 11 - RF-08

ID Requisito	RF-08	Tipo	Funcionalidad
Evento / Caso de uso	Selección del nivel de rol: Alumno.		
Descripción	Al seleccionar el rol de Alumno, se carga la opción de envío de feedback al desarrollador.		
Justificación	Es necesario en las fases iniciales de la aplicación contar con las opiniones de mejora de los usuarios, concretamente en una institución es más necesario aún.		
Criterio de cumplimiento	Una vez seleccionada la opción, se carga el menú perteneciente al rol Alumno, donde se carga dicha opción.		
Prioridad	Alta	Conflictos	#

Tabla 12 - RF-09

ID Requisito	RF-09	Tipo	Funcionalidad
Evento / Caso de uso	Selección del nivel de rol: Becario.		
Descripción	Al seleccionar el rol de Becario, el menú es cambiado para una vista más global de la aplicación.		
Justificación	El Becario puede crear nuevos usuarios, comprobar el estado de la aplicación y el servidor, y realizar tareas de mantenimiento.		
Criterio de cumplimiento	Una vez seleccionada la opción desde la ventana de login, el Becario accede a un menú específico para la administración correcta de la aplicación y el servidor.		
Prioridad	Alta	Conflictos	#



Tabla 13 - RF-10

ID Requisito	RF-10	Tipo	Funcionalidad
Evento / Caso de uso	Actualización del estado de las incidencias automáticamente sin interacción del alumno.		
Descripción	Se debe poder actualizar el estado de las incidencias del usuario cada vez que el usuario acceda a la aplicación.		
Justificación	El alumno ha de disponer del estado de sus incidencias, cada vez que inicie la aplicación.		
Criterio de cumplimiento	Cada vez que se inicie la aplicación, se actualizarán el estado de las incidencias.		
Prioridad	Media	Conflictos	#

Tabla 14 - RU-02

ID Requisito	RU-02	Tipo	Usabilidad
Evento / Caso de uso	Actualización del estado de las incidencias manualmente.		
Descripción	Se debe poder actualizar el estado de las incidencias del usuario de forma manual y en cualquier momento mediante un botón en la aplicación.		
Justificación	El usuario debe poder consultar los últimos movimientos sobre su incidencia en cualquier momento por lo que debe poder actualizar cuando quiera.		
Criterio de cumplimiento	Al pulsar el botón “Actualizar” en el menú de incidencias, la aplicación se actualiza con las nuevas actividades.		
Prioridad	Media	Conflictos	#

Tabla 15 - RU-03

ID Requisito	RU-03	Tipo	Usabilidad
Evento / Caso de uso	Descargar incidencias comprimidas del servidor.		
Descripción	Cuando una de las opciones de compresión está activada, al actualizar el estado de las incidencias se pide al Proxy que los descargue y se descargan entonces del Proxy.		
Justificación	Esta es la opción que permite que las incidencias se descarguen comprimidas.		
Criterio de cumplimiento	Si una opción de compresión está activada al actualizar la conexión se establece con el servidor que manda las incidencias comprimidas.		
Prioridad	Alta	Conflictos	#



Tabla 16 - RU-04

ID Requisito	RU-04	Tipo	Usabilidad
Evento / Caso de uso	Los campos que son rellenados para generar una incidencia han de ser validados.		
Descripción	Cuando se introduce el nombre, tipo y descripción de una incidencia, han de ser claros y concisos.		
Justificación	Una descripción ha de ser clara y escueta, los becarios han de ser capaces de resolverlas en tiempo y espacio. Ganamos en rendimiento al no sobrecargar el servidor.		
Criterio de cumplimiento	Al escribir varios caracteres el contador disminuye en función de los caracteres escritos y cambiar de color según la descripción.		
Prioridad	Medio	Conflictos	#

Tabla 17 - RU-05

ID Requisito	RU-05	Tipo	Usabilidad
Evento / Caso de uso	Visualización de estadísticas con información sobre los bytes consumidos con y sin compresión.		
Descripción	El administrador debe poder consultar unas estadísticas sobre cuantos bytes están en uso, e intercambiados entre clientes – servidores. Analizar flujo de datos entre distintos dispositivos.		
Justificación	Con el fin de comprobar que la aplicación funciona y el ahorro es significativo para el usuario, debe poder consultar los datos relativos a los bytes utilizados.		
Criterio de cumplimiento	Desde el servidor de aplicaciones Google App Engine, pueden accederse a unas estadísticas de compresión paralelas a las obtenidas por la aplicación.		
Prioridad	Alta	Conflictos	#

Tabla 18 - RU-06

ID Requisito	RU-06	Tipo	Usabilidad
Evento / Caso de uso	Utilización de la aplicación por un usuario no registrado.		
Descripción	Al iniciar la aplicación, se llevará a cabo una criba con la base de datos de usuario y el rol seleccionado.		
Justificación	Evitar el acceso a la aplicación por usuarios malintencionados.		
Criterio de cumplimiento	El acceso correcto a la aplicación se realiza si “Usuario / Password introducido” = “Usuario / Password almacenado”		
Prioridad	Alta	Conflictos	#



Tabla 19 - RF-11

ID Requisito	RF-11	Tipo	Funcionalidad
Evento / Caso de uso	Al actualizar las incidencias, si una incidencia fue previamente descargada, se reutiliza a modo de caché, a pesar de que se cierre la aplicación.		
Descripción	Cada incidencia descargada se almacena de forma persistente. Cada incidencia a utilizar se comprueba si fue descargada anteriormente buscando en la lista de descriptores de incidencias y si existe se carga de la caché, ahorrando los datos utilizados por la aplicación.		
Justificación	El ahorro de datos en dispositivos móviles representa una gran baza de uso para las aplicaciones.		
Criterio de cumplimiento	Cuando se actualiza la interfaz con incidencias de usuarios que se han visualizado previamente la imagen no se descarga del servidor.		
Prioridad	Media	Conflictos	#

Tabla 20 - RF-12

ID Requisito	RF-12	Tipo	Funcionalidad
Evento / Caso de uso	El servidor comprime las incidencias descargadas.		
Descripción	Cuando el servidor recibe una petición para descargar incidencias este las comprime antes de mandarlas al usuario. Si está activada esta opción.		
Justificación	Para ahorrar datos es necesario que el servidor los comprima antes de ser mandados al cliente.		
Criterio de cumplimiento	Cuando el cliente hace una petición al servidor este automáticamente comprime y manda las incidencias. El cliente recibe las incidencias comprimidas		
Prioridad	Alta	Conflictos	#

Tabla 21 - RF-13

ID Requisito	RF-13	Tipo	Funcionalidad
Evento / Caso de uso	El servidor almacena los datos relativos a las peticiones del cliente.		
Descripción	Cada vez que el cliente hace una petición el servidor almacena el identificador de la incidencia, así como la urgencia a ser tratada y el identificador unívoco del alumno reflejado en el dispositivo.		
Justificación	Para poder generar las estadísticas es necesario almacenar los datos de las incidencias. Una de las bazas de las aplicaciones más funcionalidades es su poca ocupación, para esta finalidad, almacenamos la mayor cantidad de información en el servidor.		
Criterio de cumplimiento	En cada petición del cliente el servidor almacena los datos en una BBDD.		
Prioridad	Alta	Conflictos	#



Tabla 22 - RF-14

ID Requisito	RF-14	Tipo	Funcionalidad
Evento / Caso de uso	El servidor procesa los datos de las estadísticas antes de mandarlos al administrador.		
Descripción	El servidor calcula en base a los datos almacenados el número de incidencias generadas por cada dispositivo, fecha y hora de las mismas.		
Justificación	Las estadísticas serán descargadas por el administrador en cualquier momento, la computación asociada a las estadísticas se realiza en el servidor para descargar al administrador y ahorrar batería.		
Criterio de cumplimiento	Cuando se solicitan las estadísticas estas llegan procesadas del servidor.		
Prioridad	Media	Conflictos	#

Tabla 23 - RF-15

ID Requisito	RF-15	Tipo	Funcionalidad
Evento / Caso de uso	La aplicación pide un número concreto de incidencias.		
Descripción	La aplicación solicita las incidencias disponibles desde la última incidencia que recibió.		
Justificación	Con esto se consigue ahorrar datos, evitando la descarga innecesaria de incidencias descargadas previamente.		
Criterio de cumplimiento	Al realizar una petición no se descargan incidencias existentes.		
Prioridad	Media	Conflictos	#

Tabla 24 - RU-07

ID Requisito	RU-07	Tipo	Usabilidad
Evento / Caso de uso	El usuario instala la aplicación.		
Descripción	El usuario (alumno o becario) descarga el apk que es provista por la universidad y la instala en su dispositivo.		
Justificación	La aplicación ha de ser provista por la universidad en un repositorio al que pueden acceder los usuarios interesados en utilizar su funcionalidad.		
Criterio de cumplimiento	Un usuario accede al repositorio, se descarga e instala el apk sin incidencias.		
Prioridad	Alta.	Conflictos	#



Tabla 25 - RU-08

ID Requisito	RU-08	Tipo	Usabilidad
Evento / Caso de uso	El usuario dispone de un dispositivo compatible.		
Descripción	La cantidad de dispositivos existentes, junto a todas las versiones de Android, hace que no podamos asegurar el completo funcionamiento de la aplicación en dispositivos incompatibles.		
Justificación	La aplicación está desarrollada en la última versión de Android, podrá funcionar en todos los dispositivos actuales a partir de cierta versión.		
Criterio de cumplimiento	Instalación y uso satisfactorio de la aplicación, sin errores.		
Prioridad	Alta.	Conflictos	#

6.DISEÑO

En este apartado, explicaremos el diseño del sistema que vamos a desarrollar, teniendo como objetivo definir la arquitectura y componentes del sistema que implementaremos.

El estudio en el apartado “Estado Actual del Mercado” ([13](#)), nos ha permitido formarnos una opinión clara y concisa, sobre las tecnologías que mejor convendrían para el desarrollo de toda la funcionalidad que nuestra aplicación requiere.

6.1 Lenguaje de programación

Buscando la mejor relación entre diseño, funcionalidad y usabilidad, se han valorado distintos lenguajes de programación para cada uno de los subsistemas de los que está compuesto este proyecto (servidor + aplicación móvil).

Después de la investigación, y posterior estudio, se ha llegado a la elección de dos lenguajes de programación basándonos en la compatibilidad con distintas bibliotecas y la necesidad de una correcta comunicación entre ambos componentes (servidor + aplicación móvil).

6.1.1 Aplicación móvil

El lenguaje de programación elegido para la aplicación móvil es el sistema operativo Android, desarrollado casi en su totalidad en Java, el SDK ofrecido por Google para Android se sirve de este lenguaje.

Para comunicar correctamente el servidor con la parte del cliente, el lenguaje de programación usado en Android será Java.

Java es un lenguaje de alto nivel orientado a objetos. El código Java en Android tiene la particularidad de no ejecutarse de la manera habitual ya que en Android no existe una JVM como tal. En Android el código se compila en un ejecutable Dalvik, que se trata de una máquina virtual especializada para Android y optimizada para dispositivos móviles.



6.1.2 Servidor

El servidor web actúa de manera completamente transparente al usuario, sin que este tenga interacción con el mismo. No es necesario diseñar una interfaz para dicho servidor, reduciendo los posibles lenguajes de programación a Java, Python y Go, al elegir Google App Engine.

6.2 Base de datos

Tanto la parte cliente como la parte del servidor, necesitan una base de datos para mantener la persistencia de datos.

6.2.1 Aplicación móvil

Android, incorpora por defecto soporte para la base de datos SQLite, se comporta de manera dinámica para dispositivos móviles que operan bajo este sistema. No requiere instalación ni configuración, al encontrarse integrada en Android.

SQLite posee las dos grandes ventajas de ocupar poco espacio y almacenar los datos en campos dinámicos, amoldándose al tamaño real de los mismos.

En la aplicación móvil se necesitan almacenar dos conjuntos de datos:

- En primer lugar, necesitamos almacenar los datos de acceso (becario-administrador y alumno). Cuando un sujeto intente acceder a la aplicación, compararemos los datos introducidos por el usuario con los contenidos en la base de datos. De esta manera, aseguraremos un acceso controlado a nuestra aplicación, a través de una criba de usuarios al inicio.
- Por otro lado, necesitaremos almacenar las incidencias generadas por el usuario. De esta manera, lograremos manejar un histórico de las mismas y una mayor facilidad en la interacción entre el usuario – aplicación.
- En un futuro, valoraremos el almacenar otras variables que pueden acelerar la interacción usuario-aplicación-servidor, como la generación automática de estadísticas de compresión y uso.

Ambas serán creadas a través de la base de datos SQLite, integrada en Android. La diferencia principal de las tablas mencionadas radica en los atributos de las tuplas que las contienen.



Estas serán las tuplas y sus atributos para nuestra base de datos:

USUARIO (nombre, contraseña)

- **Usuario:** Único para cada usuario que accede a nuestra aplicación. En caso de los alumnos, será el NIA. En caso de los becarios, será el nombre seguido de su apellido o también su NIA.
- **Contraseña:** Único para cada usuario.

La tupla nombre-contraseña, únicamente pertenecerá a un usuario, por lo que el acceso automáticamente cribará el acceso (seleccionando para ello un rol en la pantalla de acceso.)

INCIDENCIA (num, id, nombre, descripcion , estado)

- **Num:** Único para cada tipo de mensaje enviado al servidor, de esta manera el servidor efectuará una determinada acción u otra.
- **Identificador:** Único para cada incidencia, se calcula empleando la función UUID dentro del código. Clave primaria que identificara unívocamente cada incidencia.
- **Nombre:** Nombre asignado a la incidencia, por el creador de la misma.
- **Descripcion:** Explicación breve y concisa del alumno sobre la incidencia creada.
- **Estado:** Posee un valor binario, toma el valor 1 cuando ha sido resuelta y 0 en caso negativo. Es cambiada al ser recibida por parte del servidor en la parte del cliente.

Actualmente todas las incidencias son almacenadas en local para un mayor acceso a la información de manera individual, de esta manera cada usuario almacenará en su dispositivo sus incidencias. El servidor manejará un histórico de todas las incidencias para generar gráficas de evolución y estudio. Así como manejar las incidencias entre los becarios, encargados de resolverlas y los usuarios que las generan.

Para un futuro, sería evaluable implementar un “wipe” de manera local, que al cabo de un mes borrara las incidencias almacenadas en el dispositivo con la finalidad de la reducción de espacio.

También se valoraran numerosas mejoras a lo largo de este documento.

Las operaciones básicas que se realizan con esta BBDD utilizando una interfaz que simplificará la sintaxis de SQL en una clase llamada *UsuariosSQLiteHelper*.

Sobre la base de datos, emplearemos cuatro operaciones básicas:

- **Insert():** Comando asociado a “Create”, permite insertar una o más filas en la base de datos.
- **Query():** Comando asociado a “Read”, realiza una búsqueda y obtiene como resultado una fila que concuerde con los criterios de búsqueda especificados en la misma
- **Update():** Esta sentencia permite la actualización de uno o varios registros.
- **Delete():** Esta sentencia permite la eliminación de los registros estipulados.

Al no usar directamente SQL incrementamos la seguridad de nuestra aplicación (una declaración SQL es sensible de recibir ataques) e incrementamos el rendimiento, al no tener que ejecutar declaraciones SQL externas de manera repetida.

Cuando utilizamos esta capa de abstracción, simplificamos el uso y generamos menos errores de compilación.

6.2.2 Servidor

Google App Engine, usa los “datastores” de Google para almacenar grandes cantidades de datos de manera eficiente para su uso con aplicaciones de características escalables.

Dicho servidor actúa de manera transparente al usuario, sin que este tenga interacción con el mismo. La mínima interacción, es llevada a cabo por el administrador-becario cuando realiza labores administrativas desde la aplicación, pudiendo suspender el servicio.

Está diseñado para permitir a las aplicaciones interactuar con un alto volumen de datos, manteniendo un alto rendimiento.

El Google App Engine Datastore carece de esquemas y tiene las siguientes características:

- Transacciones atómicas.
- NoSQL.
- Sin tiempo de inactividad planificado.
- Alta disponibilidad en lecturas y escrituras.
- Fuerte consistencia para lecturas y antiguas consultas.

App Engine Datastore utiliza una arquitectura distribuida que gestiona automáticamente grandes cantidades de datos y su escalabilidad. La interfaz ofrece características de las bases

de datos tradicionales, pero diferente forma de describir las relaciones entre las tuplas, sus atributos y los objetos de datos.

Los tipos de búsqueda que se pueden realizar con los Datastores son más restrictivos.

En la base de datos del servidor almacenaremos el id de la incidencia junto a su nombre y descripción, para ser manejadas de manera unificada entre los roles característicos (alumno-becario-alumno || administrador).

Así pues se tendrá el siguiente diseño:

compData	
PK	<u>id</u>
	user nombre descInci estado

Ilustración 24 - Server Database

Donde los datos significan lo siguiente:

- **id**: será la clave primaria que almacenará el identificador de la incidencia, generado desde el cliente y almacenado en el servidor.
- **nombre**: Nombre de la incidencia.
- **user**: identificador del usuario (NIA). Podemos valorar este campo como clave alternativa.
- **descInci**: breve descripción de la incidencia.
- **estado**: estado actual en el que se encuentra la incidencia.
 - 1-> Incidencia abierta.
 - 0-> Incidencia cerrada

6.3 Comunicaciones

La comunicación entre los subsistemas del proyecto se realiza de manera distribuida. El servidor, al ser un PaaS (explicado en secciones anteriores), nos sirve para establecer una comunicación bidireccional aplicación – cloud (nube).

La comunicación se realiza de manera distribuida, y el servidor en este caso, no se auto-gestiona automáticamente como un ente independiente.

“Cloud” (la nube), corresponde a un sistema distribuido en el que intervienen múltiples servidores, procesando y repartiendo la información, junto a un alto manejo de la escalabilidad.



Ilustración 25 - Server PaaS



Los distintos dispositivos, donde la aplicación se encuentra instalada, se comunican con la nube. La nube, a su vez, está compuesta de varios servidores distribuidos.

Por parte de la aplicación, se pueden enviar los siguientes mensajes:

- **Mensaje de envío de incidencias:** Este mensaje se compone de X Tokens que permitirán al servidor realizar la inclusión de incidencias en la nube y el identificador de una incidencia para solicitar las incidencias disponibles a partir de ese identificador. El mensaje partirá de la estructura sobre la tupla anteriormente descrita (id, nombre, tipo, descripción, estado) y se transformará en un mensaje xml capaz de enviarse y ser traducido por el servidor.
- **Mensaje de solicitud de incidencias:** en este mensaje se recibe el ID del becario que solicita las incidencias abiertas y el servidor busca todos las incidencias abiertas (estado 1). Formará el mensaje con las incidencias pedidas y serán enviadas oportunamente al cliente.

Una vez la incidencia es descargada, se muestra en forma de tabla. Cuando es resuelta vuelve a enviarse al servidor con el valor "solved" cambiado a 0, de manera que ya no será descargada al solicitarse de nuevo las incidencias abiertas.

En la página siguiente se puede apreciar el diagrama de secuencia para las peticiones.

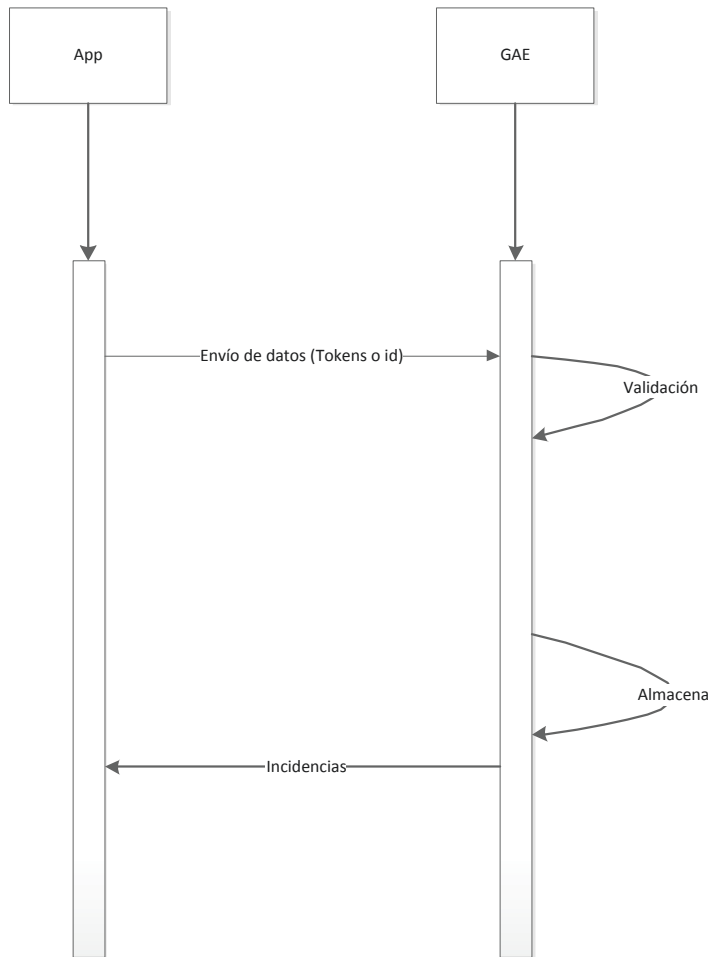


Ilustración 26 - Diagrama secuencia

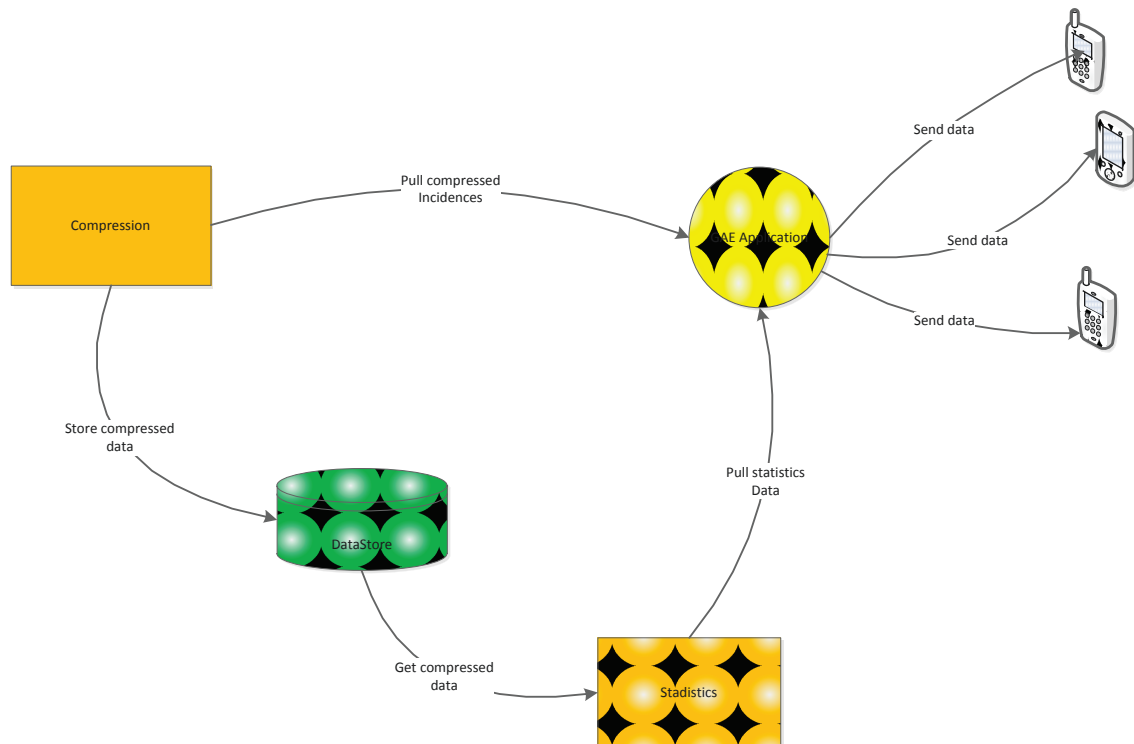


Ilustración 27 – Componentes aplicación

Aquí se explica cada componente del servidor:

- **Statistics:** recibe un identificador de incidencia y recoge los datos de la misma para procesarlos en forma de estadísticas al Smartphone. Estas estadísticas son almacenadas en forma de log dentro de una carpeta determinada de la aplicación.
- **DataStore:** Encargado de guardar los datos relativos a una petición, dichos datos son: el identificador de la incidencia, el nombre de la misma, una breve descripción y el estado (1 / 0). El identificador, al ser considerado como clave primaria, no puede repetirse e identifica unívocamente a cada incidencia.
- **Smartphones:** son los dispositivos móviles con la aplicación del cliente instalada que pueden interaccionar con el servidor.
- **GAE Application:** Google App Engine (Contiene operaciones e InciController.)

6.4 Interfaz de usuario

A través de esta apartado, definiremos la interfaz de usuario de la aplicación, la cual efectuará de intermediaria entre todas las funcionalidades de la aplicación (parte cliente – bases de datos – servidor)

El servidor carece de interfaz de usuario, únicamente interaccionamos con él a través de la aplicación gráfica de gae, y en el caso del desarrollador con el código en sí mismo.

La aplicación está adaptada a la pantalla de los smartphones actuales, aprovechando los espacios y opciones para hacerla lo más atractiva y funcional posible.

Las pantallas que podemos apreciar en el esquema de la imagen, se corresponden con cada uno de los “Activities” definidos en el diseño de la aplicación Android.

6.4.1 Manejo de incidencias Activity

Encargada de generar las incidencias a través de los siguientes campos:

- Nombre de la incidencia.
- Tipo
- Breve Descripción

La interfaz tiene un diseño minimalista, al terminar de rellenar los campos desde la aplicación podemos proceder a enviar – almacenar la incidencia, o bien borrar los campos.

La incidencia es generada automáticamente en la base de datos con el valor 0, en su status de resolución (no resuelta) y un UUID único que permite distinguirla de las demás. Una vez que la incidencia es pedida por el becario, consultada y resuelta, el valor de status cambia a 1.



6.4.2 Validar Usuario Activity

Esta Activity se encarga de cribar el acceso a la aplicación, a través de los siguientes credenciales:

- **Identificador:** Nombre de acceso provisto a los usuarios de la aplicación según su rol
- **Rol:** Puede ser becario o alumno. Es imprescindible seleccionar un rol para acceder a la aplicación, y que sea consecuente con el rol que el usuario tiene. En caso contrario, no permitirá el acceso.
- **Password:** Contraseña generada por el becario-administrador dentro de la aplicación y almacenada en la base de datos.

Una vez se introducen los datos de acceso, se comparan con los contenidos en la base de datos de la aplicación y concedemos el acceso o no.

El acceso a la aplicación se realiza comparando el usuario y password, junto con el rol seleccionado vs usuario y password almacenado en la base de datos con el rol determinado para ese usuario.

Cuando se ha concedido el acceso, el menú varía de un rol a otro:

- **Becario** – Puede realizar las siguientes acciones dentro de su menú de usuario:
 - **Ver incidencias abiertas:** Pedir al servidor que mande al cliente las incidencias cuyo estado se encuentre a 1, aún no resueltas.
 - **Estadísticas de compresión y uso:** Generación de un log dentro de la estructura del cliente donde se generarán estadísticas de todo el uso y carga de la aplicación.
 - **Registrar nuevo usuario:** Deriva a una nueva pantalla en la que, introduciendo el nombre de usuario y contraseña pertinentes, se registrará un nuevo alumno en la aplicación.
 - **Acciones globales aplicación:** Permite acciones administrativas complejas con el servidor (parar el servidor para realizar labores de mantenimiento, actualizar las credenciales del servidor y realizar un borrado de archivos antiguos y logs de incidencias, entre otras.)



- **Alumno** - Puede realizar las siguientes acciones dentro de su menú de usuario:
 - **Ver el estado de mis incidencias:** Pide al servidor el estado de sus incidencias creadas. El identificador único generado para cada incidencia se almacena de manera local en el cliente y remota en el servidor (una gran base de datos), por lo que el cliente puede disponer de ellas cuando requiera.
 - **Crear incidencias:** La incidencia se crea y cumplimenta a través de los campos nombre, tipo (hardware, software y otros), así como una breve descripción que la acompaña. Internamente se genera un identificador único que la distingue.
 - **Enviar feedback:** Al hacer uso de esta acción se despliega el correo dentro del sistema operativo Android, que permite enviar un email con las impresiones y errores destinados a mejorar la aplicación.

6.4.3 Manejo de incidencias Activity

Esta actividad está encargada de captar los campos nombre, tipo y descripción de la pantalla de creación de incidencias.

Internamente asignará un id único a la incidencia creada, la almacenará en la base de datos y enviará campos al servidor.

6.4.4 Menu

Encargado de las gestiones y acciones propias del administrador-becario

6.4.5 Menu2

Encargado de las gestiones y acciones propias del alumno

6.4.6 Registrar Usuario

Desde el menú de administrador podremos hacer uso de esta acción, que registrará nuevos alumnos en la aplicación



6.4.7 Generar estadísticas de compresión y uso.

A través de un menú simple, podremos pedir dichas estadísticas al servidor y después analizarlas cuando son almacenadas en el log dentro del cliente.

6.4.8 Acciones globales de aplicación.

El administrador podrá suspender el servidor, para realizar labores de mantenimiento, así como generar y ver la carga del mismo.



7.DESARROLLO DEL CLIENTE

Los requisitos (40) anteriormente descritos nos permiten deducir estas clases y el diagrama de flujo.

Posteriormente emplearemos los requisitos anteriormente elaborados, junto a las clases generadas, para disponer de una matriz de trazabilidad que pruebe estas clases.

En esta sección se explica la implementación de este proyecto, tanto de la aplicación móvil como del servidor.

7.1 Aplicación móvil

El desarrollo de la aplicación se realiza siguiendo los esquemas descritos durante el diseño. En esta sección se explica el desarrollo de cada componente del diseño de la aplicación con la inclusión de algunos puntos explicativos más que necesitan ser desarrollados. Los componentes son los siguientes:

- AsyncTask
- ValidarUsuario
- Menu (Menu1 & Menu2)
- MensajesIncidencia
- MainActivity
- ObjetoGlobal
- ManejoIncidencias
- BDHelper
- RegistrarUsuario
- AndroidManifest
- XML



7.1.1 AsyncTask

La utilización de la clase AsyncTask, será recursiva a lo largo de todo el proyecto, de ahí la explicación.

Todas las operaciones referentes al cliente web, se realizan en un hilo distinto al ejecutado en la interfaz de usuario.

La implantación de esta premisa, es obligatoria desde la versión 4.0 de Android, evita la ralentización de la interfaz (el temido lag), debido a la ejecución recursiva de un producto de duración variable.

Esto puede hacerse de distintas formas por medio del lenguaje Java, pero Google recomienda el uso de una clase específica en Android para tal práctica: AsyncTask.

La clase AsyncTask se encarga precisamente de esto, realizar operaciones en segundo plano y publicar los resultados en el subproceso de la interfaz de usuario, sin tener que manipular los hilos. No constituye un marco genérico para threading, y es ideal para utilizar en operaciones de corta duración (unos pocos segundos como máximo)

Para hacer uso de esta clase, se crea una subclase donde la vayamos a usar, y esta subclase heredará de AsyncTask, lo que nos permite sobrescribir varios métodos:

- Método `doInBackground(Void parameters)`
- Método `onPreExecute()`
- Método `onPostExecute(String result)`

La parte del código que necesitamos sea ejecutado en segundo plano, se incluye en `doInBackground()`, el resto de métodos son usados para otras operaciones similares, ejecutadas en el hilo principal.

7.1.2 ValidarUsuario

ValidarUsuario se encarga de autenticar a un usuario que introduce sus datos de usuario. Esta funcionalidad se codifica en “/src/com/uc3m/activity/ValidarUsuario.java”.

Primeramente, cargamos la interfaz de código localizada en el siguiente XML: “/res/layout/activity_main.xml”, que lanzará la parte de código necesario para que muestre la interfaz de acceso. Muestra un botón que al pulsarse inicia el método **comprobarAcceso()** que inicia el siguiente pseudocódigo:

- Crea un almacén para el Nick introducido, a través de EditText, userIntroducido.
- Crea un almacén para el password introducido, a través de EditText, passIntroducido.
- Crea una selección para el RadioGroup, en el que se cribará los usuarios entre becarios y alumnos.
- Recuperamos userAlmacenado / passAlmacenado de la BDHelper destinada a usuarios.
- Compara userIntroducido con userAlmacenado y el radiobutton seleccionado (becario o alumno)
- Compara passIntroducido con userAlmacenado y el radiobutton seleccionado (becario o alumno)
- Concedemos acceso si:
 - userIntroducido = userAlmacenado -> determinado rol.
 - passIntroducido = passAlmacenado -> determinado rol.
- En caso contrario se muestra un toast con el campo introducido inválido.
- En caso de no seleccionar un rol determinado, se muestra un toast con la advertencia.

Las credenciales almacenadas en BDHelper nos aseguran la unicidad entre usuario – password, de esta manera aseguramos que las credenciales introducidas se comparan con las correctas introducidas por el propio desarrollador / administrador, de la aplicación.

En futuras versiones, mantendremos la línea de cifrado sobre la base de datos para evitar substracciones de identidad.



7.1.3 Menu

Una vez realizado el acceso seguro, nos encontramos con el menú de la aplicación. Como explicamos anteriormente, el menú mostrado no es el mismo para un becario y alumno.

Las acciones administrativas son propias del menú de becario – administrador, mientras que el alumno no ha de tener constancia de esto.

Este menú está destinado a los becarios-administradores, mientras que el menú2 será determinado para los alumnos.

Las “Application” en Android, son utilizadas para mantener un estado global de la aplicación, de manera que nos sirve para poseer un punto en común donde almacenar la información (como un repositorio que vamos llenando), usado a lo largo de distintas clases.

El menú de la aplicación se encuentra en “/src/com/uc3m/activity/menu.java”. Primeramente, cargamos la interfaz de código localizada en el siguiente XML:

- “/res/layout/accesoadministrador.xml”: En el caso de que el rol de acceso sea un administrador.

En el caso de acceso seguro y basado en nuestras credenciales, no almacenamos esta variable en Application, ya que el recordatorio de las credenciales no nos aseguraría un acceso seguro a la aplicación. Nuestro nicho de usuario para ejecución de la aplicación, se basa en accesos puntuales a las mismas, de ahí la inutilidad del almacén de credenciales.

El menú se compone de las siguientes acciones globales, procederemos a explicar su división dentro de cada una de las mismas:

- “/src/com/uc3m/activity/menu.java

El método **incidenciasAbiertas()** funciona de la siguiente forma, a través de la siguiente opción del menú:

- Ver incidencias abiertas:
 - Llama a “mostrarincidencias.xml”
 - Montamos el layout pertinente, para mostrar las incidencias por pantalla.
 - Hacemos petición para descargar las incidencias.
 - Se construye el mensaje (num, id, nombre, tipo, descripción, estado, lista). Donde num es el dígito destinado a pedir las incidencias al servidor. El servidor entiende por este campo num, el tipo de mensaje y la acción que ha de realizar.
 - Son descargadas.

- Las incidencias son descargadas siguiendo la siguiente estructura (num, id, nombre, tipo, descripción, estado, lista). Concretamente en forma de lista, en el apartado correspondiente.
 - Los datos descargados, son guardados en un objeto a través de una interfaz común a servidor y cliente.
 - Dichas tuplas enviadas por el servidor, son entendidas por esta clase y mostradas en el layout.
 - Mostramos
 - Si NO son descargados, generamos excepción y mostramos *Toast pertinente*.
 - Resuelve la incidencia:
 - Nuevo estado -> 0
 - Construye el token (id, nombre, descripción, estado)
 - Actualiza
- **Registrar nuevo usuario.**
- Pasamos a RegistrarUsuario activity: En esta actividad se dispondrán gráficamente los campos a rellenar para el nuevo usuario, seguidamente se almacenarán en la base de datos y volveremos al menú principal.
 - Montamos la interfaz layout “registronusuario.xml”: Encargada de mostrar los campos Nick y Nombre a rellenar, junto con los botones guardar y borrar.
 - Relleno de campos Nick / Rol / Password: Capturados como texto editable.
 - Almacenamos los datos en BDHelper como nuevo usuario creado: Posteriormente insertamos a través de las sentencias Mysql en la base de datos, los campos adquiridos anteriormente.
 - Regresamos a menú.xml
- **Acciones globales de usuario.**
- Pasamos a Acciones Globales activity.
 - Montamos el layout accionesglobales.xml
 - Si el administrador decide suspender el servicio.
 - Construimos el mensaje con el identificador pertinente.
 - Enviamos el mensaje al servidor.
 - En este momento, el servidor entra en suspensión y todas las nuevas incidencias enviadas serán respondidas con un mensaje pertinente.
 - Si el administrador decide reponer el servicio.
 - Construimos el mensaje con el identificador pertinente.
 - Enviamos el mensaje al servidor.
 - En este momento el servidor vuelve a ser operativo.



7.1.4 Menu2

- `"/src/com/uc3m/activity/menu2.java`

Esté menú está construido específicamente para el alumno, con las funciones pertinentes que un alumno puede desempeñar en nuestra aplicación.

El método **enviarIncidencia()** funciona de la siguiente forma, a través de la siguiente opción del menú:

- **Crear incidencias:**
 - Acción Intent hacia `"manejodeincidencias.java"`
 - En este layout, se podrán rellenar como editText los campos (nombre, tipo, descripción)
 - Almacenamos nombre de incidencia a través de EditText, hacia variable nombre.
 - Los campos anteriormente descritos son almacenados en nuestra base de datos global.
 - Almacenamos descripción de incidencia a través de EditText2, hacia variable descripción.
 - Los campos anteriormente descritos son almacenados en nuestra base de datos global.
 - Generamos token (*id, nombre, descripción*)
 - Para identificar unívocamente a una incidencia hacemos uso de la librería UUID que genera un id único cada vez que se ejecuta y asignamos uuid -> id.
 - Subimos a GAE
 - Componemos el mensaje (num, id, nombre, tipo, descripción, estado)
 - El num = número acordado con el servidor para indicarle que estamos mandando una incidencia.
 - Estado:
 - 1 -> Abierta
 - 0 -> Cerrada
 - Almacenamos id en BDHelper
 - Además almacenamos (insert (campo, "campo")) en la base de datos la incidencia.

Método **recuperarIncidencia()**, a través del id almacenado en la base de datos procedemos a recuperar la incidencia anteriormente generada:

- Recuperamos id en BDHelper
 - El id único generado con UUID se almacena en la base de datos, junto con el resto de datos de incidencias. Recuperamos este campo.
- Hacemos petición para descargar valores asociados a id
 - Mandamos el mensaje construido (num, id, nombre, tipo, descripción, estado, lista) donde num = dígito acordado con el servidor que muestre nuestra intención de recuperar la incidencia asociada a id.
- Son descargadas
 - El servidor responde con la misma estructura de mensaje que compusimos al mandar la incidencia, con los campos rellenos de dicha incidencia. Teniendo en cuenta, el estado de las incidencias 1 / 0.
- Mostramos
 - Los campos recuperados del servidor, son enviados al layout pertinente y mostrados por pantalla.
- En caso de que id no se encuentre -> Error mostrado “no id” en *toast*

El método encargado de enviar feedback al desarrollador posee la siguiente lógica:

- Realiza un intent hacia la aplicación mail, instalada en el teléfono.
- Automáticamente rellena los campos de contacto.
- Cuando se cierra la aplicación mail, vuelve al Menu2

7.1.5 Compresión de incidencias

- “/src/com/uc3m/activity/incidencias.java

El método **getCompressedIncidence(long)** funciona de la siguiente forma:

- Se hace un POST al servidor con el ID de la incidencia.
- Se guarda la respuesta al POST.
- SI la respuesta es OK:
 - Se hace una petición para descargar las incidencias comprimidos.
 - Son descargados.
 - Los datos descargados se guardan en un objeto a través de una interfaz común a servidor y cliente.
 - Se descomprimen los datos.



- Se deserializan los datos.
- Se devuelven las incidencias.
- SI NO se recibe OK por respuesta:
 - Lanza excepción informando.

7.1.6 Acciones Globales

- `"/src/com/uc3m/activity/accionesglobales.java`

El método **suspenderServidor(View v)** funciona de la siguiente forma:

- Montamos el layout correspondiente a `"accionesglobales.xml"`, accedidas desde Menu1 (de administración).
- Al hacer click en el botón "Suspender servicio", internamente:
 - Se construye el mensaje pertinente al servidor, donde el campo `num` = dígito que indica al servidor, tipo de mensaje -> `suspender servicio`.
 - El servidor entra en estado de suspensión hasta que reciba otro mensaje con `num` = dígito que indica al servidor la reanudación del servicio.
 - El mensaje construido (`num, id, tipo...`)
 - Todas las incidencias son rechazadas y respondidas con un mensaje de error.
 - El servidor contesta con un mensaje al administrador "OK", el servicio ha sido suspendido satisfactoriamente.
 - El propio layout construido, bloquea el botón de "Suspender" y habilita el "Reanudar"
- Al hacer click en el botón "Reanudar", internamente:
 - Se contruye el mensaje (`num, id, tipo...`), donde `num` = dígito que indica al servidor la reanudación del servicio.
 - El servidor contesta con un mensaje al administrador "OK", el servicio se ha restaurado satisfactoriamente.
 - En este momento el servidor vuelve a estar operativo.

El método **generarEstadisticas(View v)** funciona de la siguiente forma:

Este Activity es el encargado de mostrar las estadísticas descargadas del servidor. Nada más iniciarse el Activity se ejecuta un AsyncTask para pedir los datos y al descargarlos actualiza la interfaz mostrándolos en forma de gráficas.

Es importante conocer las estadísticas con el fin de detectar anomalías en el servicio, así como mejorar en futuras versiones la compresión de los datos.



La lógica es la siguiente:

- AsyncTask:
 - Se establece la URL del servidor en un ClientResource.
 - Se manda el ID del usuario con un POST.
 - Se recupera la respuesta al POST.
 - SI la respuesta es OK:
 - Se hace un GET para descargar las estadísticas.
- Se descomprimen las estadísticas.
- Se deserializan.
- Se cargan los datos en los objetos GraphView.
- Se añaden dichos objetos a la interfaz.

7.1.7 Main Activity

Este Activity es el principal de la aplicación, ejerce de tronco principal enlazando al resto de componentes de la aplicación. Transfiere desde la validación de usuario hasta los menus y ejerce de control para las variables.

La clase se encuentra en “/src/com/uc3m/activity/mainactivity.java”. Además se encargará de pintar la interfaz del menú y enlazar a las distintas opciones que en él se encuentran.

En primer lugar:

- Mandamos un nuevo intent hacia validar usuario:
 - Se capturan el nombre / usuario / rol introducido.
 - Se compraran dichos campos con los almacenados en la bd.
 - SI coinciden:
 - Realizamos un nuevo intent hacia Menu / Menu2, según represente.
 - SI NO coinciden:
 - Se niega el acceso.

7.1.8 Objeto Global

Es una clase intermedia entre todas las actividades de la aplicación, encargada de manejar variables de manera global.



7.1.9 BDHelper

Esta clase se encarga de crear la BBDD y actualizar sus parámetros si se actualiza la versión de la BBDD.

Llamamos al objeto onCreate() que ejecuta una query con los parámetros creados dentro de la base de datos.

Esta clase realiza la siguiente funcionalidad:

- Crea la tabla Alumno (nombre, contrasena)
- Crea la tabla Becario (user, pass)
- Crea la tabla Incidencia (id, nombre, tipo, descripción, estado)
- Comprueba si existen estas tablas.
 - En caso afirmativo -> no hace nada.
 - En caso negativo -> las crea
 - Comprueba el path de la ubicación donde se crea la base de datos.
 - Establece permisos para la base de datos y la crea.
- Al intentar actualizar la base de datos:
 - Se elimina la versión anterior de la base de datos.
 - Se crea la nueva versión de las tablas

Actualmente manejamos las siguientes bases de datos:

- Base de datos de Alumno:
 - Almacena los datos [nombre, contraseña]. El nombre se basa en el nia del alumno, por lo que nos aseguramos una restricción inherente a la clave primaria (no puede repetirse), que incrementa la seguridad de nuestra aplicación.
- Base de datos de Becario
 - Almacena los datos [user, pass]. El nombre del becario es único junto a su password, la cantidad de becarios es menor a la de alumnos.
- Bases de datos sobre Incidencias [id, nombre, tipo, descripción, estado]. PK id, generada a través de una funcionalidad inherente a Java que permite generar números únicos aleatorios, asegura la unicidad de la incidencia.

7.1.10 AndroidManifest

Cada aplicación debe tener un AndroidManifest.xml en su directorio raíz. El Manifest de una aplicación Android, contiene información esencial sobre la estructura de la aplicación.

Entre otras cosas, el manifest realiza las siguientes acciones:

- Nombra el paquete Java para la aplicación, que sirve como identificador único.
- Describe los componentes de la aplicación.
- Determina los procesos anfitriones.
- Declara los permisos que los demás están obligados a tener para interactuar con el resto de la aplicación.

Inicialmente, dentro del manifest, es necesario declarar los permisos que las Activities tendrán con el sistema. Actualmente, en la aplicación, tenemos concedidos los siguientes permisos:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

- El permiso, WRITE_EXTERNAL_STORAGE es necesario para el funcionamiento del almacenamiento de incidencias, que escribe en la memoria interna del teléfono.
- El permiso de INTERNET es necesario para comunicarse con el GAE.
- READ / WRITE CONTACTS es necesario para manejar la lista de contactos contenida en el teléfono, con posibles líneas futuras que adquieran las credenciales del teléfono.
- ACCESS_NETWORK_STATE Permite que las aplicaciones accedan a la información sobre redes.

También es necesario definir todos los Activities y Services, en caso contrario no funcionarán por no ser reconocidos.

El activity Inicial posee la siguiente declaración en el Manifest:

```
<activity
    android:name="com.uc3m.activity.MainActivity"
    android:label="@string/app_name" >

    <intent-filter>
```



```
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

El primer Activity que ha de ejecutarse al arrancar la aplicación queda estipulado a través de la etiqueta action.Main.

7.1.11 XML

Los XML son usados a lo largo de todo el proyecto, junto al Manifest e Interfaces que son cargadas a través del XML determinado.

Un XML, por ejemplo el de validar usuario sigue la siguiente estructura:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView3"
        android:layout_marginTop="32dp"
        android:text="Identificador"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/identificador"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="14dp"
        android:ems="10" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/borrarAcceso"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/comprobarAcceso"
```

```
        android:layout_alignBottom="@+id/comprobarAcceso"  
        android:onClick="borrarAcceso"  
        android:text="Borrar" />  
  
    <RadioButton  
        android:id="@+id/RadioButton01"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_above="@+id/textView4"  
        android:layout_alignParentLeft="true"  
        android:onClick="alumnoSeleccionado"  
        android:text="Admin" />  
  
</RelativeLayout>
```

7.1.12 Bibliotecas

A lo largo del proyecto, usamos varias APIs para posibilitar el funcionamiento de la aplicación, estas APIs se localizan dentro del directorio “/libs” del proyecto y son las siguientes:

- Commons-compress-1.4.1.jar: esta biblioteca es usada para los algoritmos de compresión de GZip.
- GraphView-3.0.jar: es la biblioteca usada para representar y organizar los datos estadísticos en forma de gráficas.
- Org.restlet.jar: es la biblioteca usada para las comunicaciones con el servidor usando el servicio RESTlet.

A parte de estas bibliotecas, será necesario disponer de las bibliotecas iniciales dispuestas por el SDK de Android en Eclipse, ya que en caso contrario se podrían producir anomalías en el desarrollo y funcionamiento de la aplicación.



8.DESARROLLO DEL SERVIDOR

8.1 Servidor

El servidor, es desarrollado siguiendo los esquemas anteriormente descritos durante el diseño. En esta sección se explica el desarrollo y análisis de cada componente del mismo, el servidor está compuesto:

- InciController
- Operaciones
- Compression
- Statistics
- DataStore
- Interfaces
- Bibliotecas

8.1.1 InciController

Nos encontramos con un símil entre la aplicación de Android y esta parte del servidor, podríamos asemejar esta parte como “Application” del servidor, ya que sirve como tronco para el resto de ramas que componen la aplicación.

En función de la cabecera del paquete enviado por el cliente, nuestro servidor puede conocer qué tipo de acción ha de realizar.



Tabla 26 - Identificador / Mensaje

Identificador	Tipo de Mensaje
1	Nueva incidencia desde cliente
2	Petición incidencias abiertas.
3	Recibir incidencias del servidor. (Cliente)
4	Labor administrativa – Suspensión.
5	Solicitud de estadísticas de compresión / uso
6	Labor administrativa – Reanudar suspensión

- Recibimos un mensaje compuesto, según la estructura definida para el intercambio.
- Analizamos la cabecera del paquete.
 - Cabecera = 1 -> El cliente está mandando una nueva incidencia.
 - Analizamos nombre, tipo, descripción, estado del paquete.
 - Nombre -> nombre incidencia generada.
 - Tipo -> tipo de incidencia.
 - Descripción -> descripción de la incidencia.
 - Estado ->
 - 1 -> Incidencia Abierta
 - 0 -> Incidencia Cerrada (Resuelta)
 - Almacenamos (nombre, tipo, descripción, estado) en la base de datos.
 - Cabecera = 2 -> El cliente está pidiendo incidencias abiertas.
 - Recuperamos de la base de datos incidencias cuyo estado = 1
 - Construimos el mensaje (nombre, tipo, descripción, estado) según la estructura definida para el intercambio.
 - Enviamos al cliente.
 - Cabecera = 3 (Estudiada en la parte del cliente)
 - Cabecera = 4 -> El becario está pidiendo la suspensión del servicio.
 - Cada mensaje con cabecera distinta a 6 (reanudar suspensión), será rechazado con un mensaje de error.
 - Cabecera = 5 -> El becario está pidiendo estadísticas.
 - Entramos en [Compression \(78\)](#)
 - Cabecera = 6 -> El becario está pidiendo la reanudación del servicio.
 - El servicio entra en estado normal.

Utilizamos el método **createInboundRoot()** para utilizarlo como enrutador entre los demás componentes de la aplicación, accedidos por el cliente. Empleamos router al crearse, añadiéndole dos aplicaciones (compression y statistics) añadiéndole a cada una apropiadamente una URI.

8.1.2 Operaciones

Encargada de realizar las operaciones pedidas por InciController para el manejo de datos desde la DataStore hasta InciController.

Posee dos operaciones: **nuevalnc** y **peticionInc**.

La lógica de **nuevalnc**, es la siguiente:

- Obtiene los datos entre el formato de mensaje xml enviados por el cliente. Adquiriendo los datos entre las etiquetas xml.
- Añade los datos y manda al DataStore para su almacenamiento
- Se procesan los datos para ser usados.

La lógica de **peticionInc**, es la siguiente:

- Recupera las incidencias almacenadas en el DataStore cuyo estado = 1
- Construye un arraylist.
- Construye el mensaje (nombre, tipo, descripción, estado, lista), con las incidencias activas recuperadas en esa lista.
- Envía el mensaje al cliente.



8.1.3 Compression

Es la clase encargada de pedir la lista de incidencias, comprimirlas y mandarlos al cliente. Contiene tres métodos asociados a POST, GET y DELETE, que son `receive(Representation)`, `retrieve()` y `remove()` respectivamente. Cabe destacar que el método por defecto, no comprime las incidencias, el método de compresión se habilita si es seleccionado.

La lógica de `receive(Representation)` es la siguiente:

- Crea una nueva `Application` y le asigna los datos existentes en ese momento.
- Asigna a la variable `tokens` del `Application` el texto recibido.

La lógica de `retrieve()` es la siguiente:

- Se crea un array de bytes vacío.
- Se reciben las credenciales del cliente con el ID de la incidencia.
- Se procesan los datos para ser usados.
- Se crea un `ConfigurationBuilder` con las credenciales.
- Se genera un objeto `Incidencia` a través del `ConfigurationBuilder`.
- Se solicitan las últimas incidencias del usuario
- Se serializa la lista de `Incidencias` recibidas.
- Se comprimen los datos serializados.
- Se guardan los datos asociados a la petición a través del `DataStore`.
- Se devuelven los datos serializados.

La lógica de `remove()` es la siguiente:

- Se recuperan los datos del `Application`.
- Se establecen a "vacío" los `tokens` del `Application`.

8.1.4 DataStore

Clase encargada de manejar los datos almacenados en Google App Datastore. Consta de dos métodos, uno destinado a guardar los datos y otro a recuperarlos.

El método empleado para guardar los datos posee la siguiente funcionalidad:

- Se reciben los datos en formato xml.
- Son limpiados de etiquetas xml, quedando solo los datos útiles
- Llamamos a añadir incidencia (id, nombre, tipo, descripción, estado)
- Rellena los campos obtenidos de la recepción del mensaje y los almacena en la base de datos.
- Se devuelve true si todo OK.
- Devuelve excepción en caso de ERROR.

El método empleado para recuperar los datos posee la siguiente lógica:

- Se crea un objeto de la interface ArrayList.
- Se crea una nueva Query para buscar en la tabla de datos.
- Se añade un filtro con el ID pasado por parámetros.
- Se ejecuta la Query.
- Se guardan los datos recibidos por la Query en el objeto ArrayList.
- Se devuelve el objeto con los datos recolectados.
- El objeto contiene los datos recolectados para su envío.



8.1.5 Interfaces

El servidor cuenta con varios “Interfaces”, al igual que en la aplicación Android, para que la comunicación por red sea lo más fluida y competente.

Estos interfaces son los mismos que explicamos anteriormente.

8.1.6 Bibliotecas

Las bibliotecas empleadas en el servidor, son las mismas descritas para la aplicación Android .

9.TIMELINE

La siguiente tabla, establece la duración de las tareas – principio – fin, con una previsión ideal

Tabla 27 - Tareas Ideal

Tarea	Principio	Fin	Tiempo
Reunión inicial con el tutor.	24/01/2014	24/01/2014	1 día
Análisis inicial del mercado.	20/02/2014	20/02/2014	1 día
Elección de herramientas y tecnologías.	25/02/2014	25/02/2014	1 día
Boceto inicial del cliente.	03/03/2014	05/03/2014	3 días
Aprobación del tutor, sobre tecnologías empleadas y boceto inicial del cliente.	11/03/2014	11/03/2014	1 día
Diseño del cliente	13/03/2014	20/03/2014	8 días
Pruebas sobre el cliente	31/03/2014	31/03/2014	1 día
Diseño del servidor	01/04/2014	07/04/2014	7 días
Pruebas sobre el servidor	08/04/2014	08/04/2014	1 día
Conexión cliente - servidor	09/04/2014	20/04/2014	12 días
Memoria	21/04/2014	15/05/2014	25 días



La siguiente tabla, establece la duración de las tareas – principio – fin, con una previsión real

Tabla 28 - Tareas Real

Tarea	Principio	Fin	Tiempo
Reunión inicial con el tutor.	24/01/2014	24/01/2014	1 día
Análisis inicial del mercado.	20/02/2014	24/02/2014	5 días
Elección de herramientas y tecnologías.	25/02/2014	25/02/2014	1 día
Boceto inicial del cliente.	03/03/2014	07/03/2014	5 días
Aprobación del tutor, sobre tecnologías empleadas y boceto inicial del cliente.	11/03/2014	11/03/2014	1 día
Diseño del cliente	13/03/2014	27/03/2014	15 días
Pruebas sobre el cliente	31/03/2014	31/03/2014	1 día
Diseño del servidor	01/04/2014	17/04/2014	17 días
Pruebas sobre el servidor	22/04/2014	22/04/2014	1 día
Conexión cliente - servidor	24/04/2014	08/05/2014	15 días
Memoria	22/05/2014	17/06/2014	27 días

9.1 Análisis

Podemos observar la dispersión entre los puntos del plazo ideal y el plazo real.

El principal motivo, para tener una cierta desviación considerable plazo ideal – plazo real, es el encuentro de otro trabajo por mi persona. La incompatibilidad de horarios hizo mella en este proyecto, como podemos observar en la comparación ideal –real.

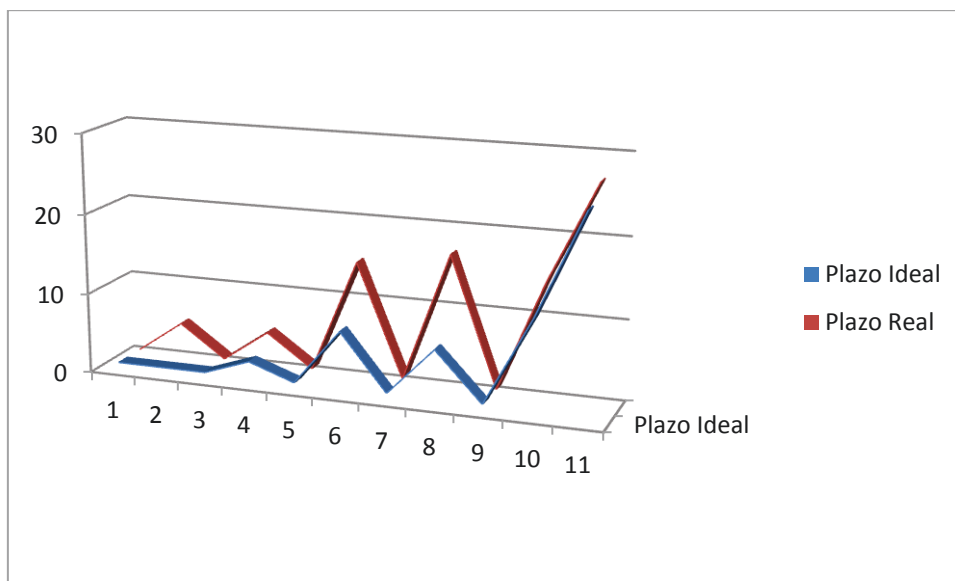


Ilustración 28 - Desviación

10. PRUEBAS DE EVALUACIÓN

En esta sección, procederemos a describir las pruebas de evaluación que realizaremos sobre la aplicación, para asegurar su buen rendimiento y fiabilidad. Para estas pruebas, emplearemos un servidor y un Smartphone, cuyas especificaciones se describen en la siguiente tabla:

Tabla 29 - Hardware de prueba

Modelo	Samsung Galaxy S I-9000
CPU	Cortex A-8 1 Ghz
Memoria	512 Mb RAM
Conectividad	HSDPA

Los recursos reales utilizados en el servidor son difíciles de calcular debido a que se trata de un sistema distribuido que asigna los recursos en función de las necesidades de la aplicación en cada momento, luego dispone de unos recursos dinámicos y adaptados a la carga de trabajo del servidor.

Vamos a realizar tres pruebas específicas por método, empleando para tal fin la funcionalidad que provee jUNIT para pruebas unitarias.



10.1 Pruebas Unitarias

Para la realización de estas pruebas unitarias, hemos utilizado la herramienta JUnit en el IDE Eclipse.

Su uso se basa en anotaciones Java:

- `@Test`: Marca un método como prueba, la herramienta lo identifica y ejecuta el proceso de prueba.
- `@Before`: El método señalado es ejecutado antes de cada uno de los marcados con `@Test`.
- `@After`: Se ejecuta después de cada `@Test`.
- `@BeforeClass`: Se ejecuta una sola vez antes de la batería de pruebas definida en la clase, el método ha de ser `static`.
- `@AfterClass`: Es ejecutado al final del proceso completo, el método ha de ser `static`.
- `@Ignore`: Marca un método o una clase completa para que no se ejecute.

Para cada método, realizaremos tres pruebas unitarias.



10.1.1 ValidarUsuario

En esta primera prueba, intentaremos acceder a la aplicación 10 veces, asignando un usuario introducido incorrecto.

Tabla 30 - Prueba Unitaria 1

Asignar userIntroducido -> incorrecto	
Número de pruebas	10

Para esta segunda prueba, intentaremos acceder a la aplicación 10 veces, con un usuario introducido correcto.

Tabla 31 - Prueba Unitaria 2

Asignar userIntroducido -> correcto	
Número de pruebas	10

En esta última prueba, intentaremos acceder 10 veces con un password introducido incorrecto.

Tabla 32 - Prueba Unitaria 3

Asignar passIntroducido -> incorrecto	
Número de pruebas	10



10.1.2 Menu

Tabla 26 - Identificador / Mensaje = cabecera que identifica el tipo de mensaje que el cliente, está enviando al servidor.

En esta primera prueba, asignaremos el idMensaje = solicitar incidencias abiertas.

Tabla 33 - Prueba Unitaria 4

Asignar idMensaje -> correcto	
Número de pruebas	10

En esta segunda prueba, asignaremos idMensaje = valor inexistente (números negativos)

Tabla 34 - Prueba Unitaria 5

Asignar idMensaje -> incorrecto	
Número de pruebas	10

La siguiente prueba, intentará registrar un usuario con valor nombre -> incorrecto

Tabla 35 - Prueba Unitaria 6

Asignar nuevoUser -> incorrecto	
Número de pruebas	10

Y para finalizar la batería de pruebas, intentaremos registrar un nuevo usuario con password incorrecto.

Tabla 36 - Prueba Unitaria 7

Asignar passNUsuarios -> incorrecto	
Número de pruebas	10



10.1.3 Menu2

En la primera prueba intentaremos generar una nueva incidencia con valores incorrectos.

Tabla 37 - Prueba Unitaria 8

Asignar nombreIncidencia -> incorrecto	
Número de pruebas	10

En esta segunda prueba generaremos una nueva incidencia con valores semicorrectos (compuestos de números, letras y signos)

Tabla 38 - Prueba Unitaria 9

Asignar nombreIncidencia -> correcto descIncidencia -> semicorrecto	
Número de pruebas	10

Asignaremos un id de mensaje correcto, que solicite las incidencias del usuario, pero con identificadores únicos incorrectos

Tabla 39 - Prueba Unitaria 10

Asignar idIncidencia -> incorrecto idMensaje -> correcto	
Número de pruebas	10



10.1.4 Mensajes Incidencias

En esta clase, se generarán y construirán los mensajes a enviar hacia el servidor. Es imprescindible, para una buena comunicación cliente – servidor, ya que indica al servidor el tipo de información que estamos enviando.

En la primera prueba, construiremos mensajes con campos incorrectos.

Tabla 40 - Prueba Unitaria 11

Asignar id -> incorrecto tipo -> incorrecto status -> incorrecto nombre -> incorrecto	
Número de pruebas	10

En esta segunda prueba, construiremos mensajes semi-correctos

Tabla 41 - Prueba Unitaria 12

Asignar id -> correcto tipo -> incorrecto status -> correcto nombre -> incorrecto	
Número de pruebas	10

Finalmente incluiremos y construiremos mensajes correctos.

Tabla 42 - Prueba Unitaria 13

Asignar id -> correcto tipo -> correcto status -> correcto nombre -> correcto	
Número de pruebas	10



10.2 Pruebas de aceptación (Cliente)

Estas pruebas, son la consecución de las pruebas unitarias por parte del programador –tester, en la parte del cliente.

Con el cliente, vamos a realizar diversas pruebas, que determinen si la aplicación satisface sus requerimientos iniciales.

Tabla 43 - Pruebas de aceptación

Nombre prueba
Se genera una incidencia.
Becario descarga incidencia
Becario notifica que la incidencia está resuelta.
Alumno notifica que su incidencia está resuelta.
Becario registra nuevo alumno.
Becario suspende el servicio.
Becario reanuda el servicio.
Alumno envía feedback al desarrollador.



10.3 Conclusiones de las pruebas de evaluación

Analizando los resultados obtenidos, podemos afirmar que el uso de la aplicación sobre GAE cumple las expectativas, ahorrando una importante cantidad de tráfico y bytes. Los tiempos al usar 3G para descargar la lista de incidencias, se asemejan a realizar la misma acción sobre la red WiFi.

Observamos que la red 3G es más lenta, algo esperable, esto nos lleva a plantearlos un mayor índice de compresión para líneas futuras teniendo en cuenta que la cobertura e influencia del resto de aplicaciones contenidas en el Smartphone pueden llevarnos a una fluidez deficiente en la respuesta de la aplicación.

Actualmente las velocidades de descarga sobre incidencias son aceptables, no ha habido contratiempos durante la realización de las pruebas.

El acceso a esta aplicación es siempre seguro, no pudiendo realizarse accesos por usuarios no autorizados (como ha quedado constatado con las pruebas), esto repercute directamente en la seguridad de la aplicación.

Comprobamos también que el ciclo de vida de una incidencia, dentro de nuestra aplicación, se realiza de manera satisfactoria.

11. PRESUPUESTO

Para realizar el cálculo de presupuestos sobre el proyecto, hemos tomado en consideración varios aspectos, en primer lugar se encontrarán los requisitos técnicos y hardware:

Tabla 44 - Presupuesto 1

Elemento	Coste unitario	Nº Unidades	Total
Ordenador Personal para el desarrollo	740,00 €	1	740,00 €
Google App Engine	0 €	1	0 €
Coste Total			740 €

Tras los costes hardware se deben tener en cuenta los recursos de software utilizados:

Tabla 45 - Presupuesto 2

Elemento	Coste unitario	Nº Unidades	Total
Licencia Windows 7	164,00 €	1	164,00 €
Licencia Microsoft Office 2013	199,00 €	1	199,00 €
Software libre [Eclipse + GAE]	0 €	1	0 €
Coste total			363,00 €

El tiempo de vida estimado de los recursos hardware, a donde los recursos software están adscritos, es de 30 meses de duración, por lo que se ajustan los costes amortizados, en base al a duración del proyecto, de 8 meses:

Tabla 46 - Presupuesto 3

Elemento	Coste completo	Coste mensual	Coste amortizado
Amortizamiento HW	840,00 €	28,00 €	224,00 €
Amortizamiento SF	363,00 €	12,10 €	96,80 €
Coste total			320,8 €

A continuación, desglosaremos los costes de personal comenzando por el propio presupuesto:

Tabla 47 - Presupuesto 4

Elemento	Coste hora	Coste mes	Horas totales	Coste proyecto
Becario (Diego)	5 €	400,00 €	480	2400,00 €
Personal investigador (Luis Miguel)	11,25 €	1800,00 €	300	3375,00 €
Personal investigador (Rafael)	11,25 €	1800,00 €	300	3375,00 €
Coste total				9150,00 €

Han de considerarse los gastos de transporte:

Tabla 48 - Presupuesto 5

Medio de transporte	Tipo de gasolina	Coste por litro	Coste mensual - Abono	Coste total
Tren	----	----	200,00 €	200,00 €



Gastos fungibles:

Tabla 49 - Gastos fungibles

Elemento	Coste hora	Coste mes	Horas totales	Gasto total
Luz	0,15 €	72,00 €	480	72,00 €
Agua	--	16,75 €	--	16,75 €
Internet	--	36,24 €	--	36,24 €
Coste total				124,99 €

Teniendo en cuenta los costes anteriormente descritos, procedemos a constatar el coste global de la siguiente forma:

Tabla 50 - Coste total proyecto

Elemento	Total
Hardware	224,00 €
Software	96,80 €
Personal	9150,00 €
Medio de transporte	200,00 €
Gastos fungibles	124,99 €
Total	8475,79 €

12. INSTALACIÓN Y CONFIGURACIÓN

En esta sección, procederemos a incluir los manuales de instalación y manejo inicial de la aplicación.

12.1 Manual de instalación del servidor

La instalación del servidor para ser utilizado será sencilla e intuitiva, al usar Google App Engine. Grandes bazas a su favor, la no necesidad de configurar el servidor, no instalar ningún plugin ni tecnología necesaria para hacerlo funcional fuera de estos parámetros y la no necesidad de instalar bases de datos. La puesta en marcha e instalación del servidor, se vuelven acciones sencillas al tener únicamente que hacer click en el botón “deploy” desde la misma interfaz de Eclipse junto al plugin GAE.

En primer lugar, explicaremos la instalación del plugin de GAE dentro de Eclipse

Eclipse, es descargado desde su página oficial de manera gratuita: <http://www.eclipse.org>.

Entre las versiones que podemos elegir para descargar, aconsejamos optar por la versión Classic o la versión Java Developers.

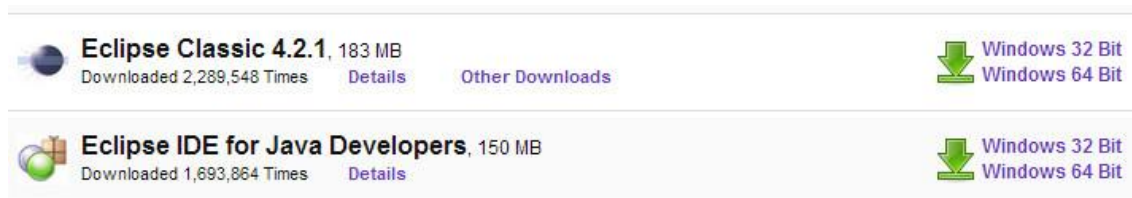


Ilustración 29 - Descarga Eclipse

Una vez descargado y descomprimido, únicamente tendremos que hacer doble click sobre el launcher eclipse, ya que no requiere instalación. Realizaremos las siguientes acciones:

- Menú Help > Install New Software

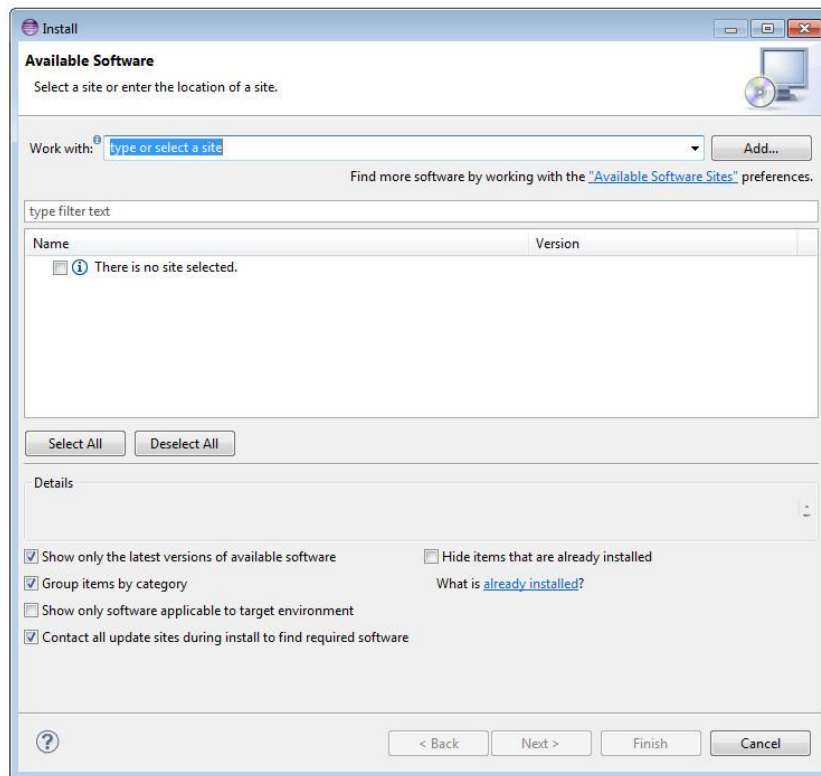


Ilustración 30 - Añadir repositorio

En el campo habilitado para introducir el acceso web a plugin (entre Work with y el botón Add, como podemos observar en la imagen), introduciremos la siguiente dirección y pulsaremos ENTER:

<https://developers.google.com/eclipse/docs/install-eclipse-4.2>

Este link incluye además las instrucciones de instalación en inglés.

Seleccionando todas las opciones que se nos muestran disponibles al terminar de realizar la búsqueda del plugin, por parte de Eclipse. Al seleccionar todas las opciones planteadas por Eclipse, instalaremos además ADE (Android Development Extensions), que nos permitirá disponer de las herramientas necesarias para el desarrollo dentro de la plataforma Android.

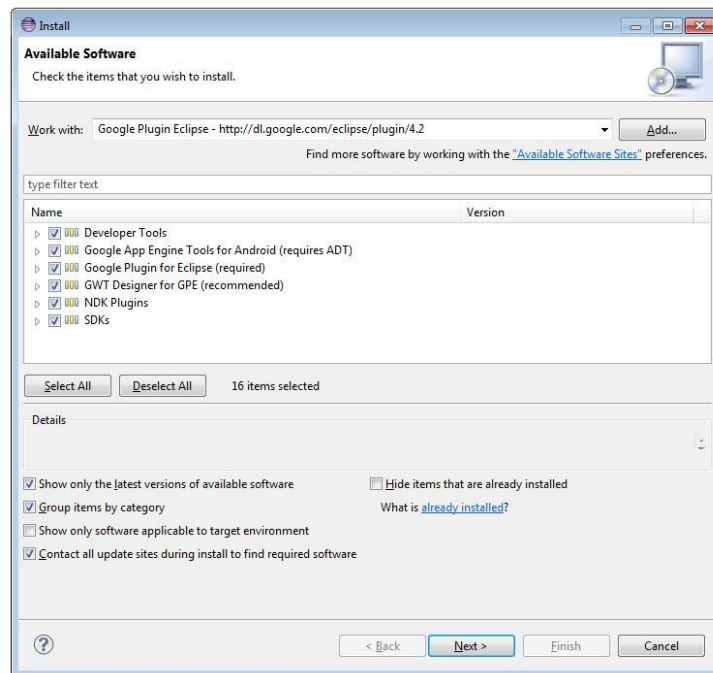


Ilustración 31 - Añadir paquetes

La descarga de paquetes, tomará su tiempo, es por lo cual recomendamos comenzar a crear una nueva aplicación dentro de la web habilitada para tal fin:

<https://appengine.google.com/start/createapp>

Si no creamos la aplicación, no podremos enlazarla con el plugin GAE anteriormente descrito, y por lo tanto no tendremos parte servidor.

Create an Application

You have 9 applications remaining.

Application Identifier:

.appspot.com

All Google account names and certain offensive or trademarked names may not be used as Application Identifiers.

You can map this application to your own domain later. [Learn more](#)

Application Title:

Displayed when users access your application.

Authentication Options (Advanced): [Learn more](#)

Google App Engine provides an API for authenticating your users, including Google Accounts, Google Apps, and you'll need to specify now what type of users can sign in to your application:

☒ Open to all Google Accounts users (default)

If your application uses authentication, anyone with a valid Google Account may sign in.

☐ Restricted to the following Google Apps domain:

e.g. foo.com

If your application uses authentication, only members of this Google Apps domain may sign in. If you

Ilustración 32 - Crear aplicación GAE

Al abrir la web para la creación de la aplicación, nos encontraremos con un formulario idéntico al de la imagen, donde tendremos que introducir el nombre de nuestra aplicación

Seleccionar, además, el título para la misma. Realizando todos estos pasos, dispondremos de una aplicación creada en Google App Engine, lista para ser enlazada con nuestro plugin anteriormente descrito y ya instalado en Eclipse.

Tras la descarga e instalación de los paquetes, por parte de Eclipse sobre la web de Google App Engine, nos encontraremos con un nuevo botón en la barra de herramientas para Eclipse, dentro del cual se encuentra “Deploy to App Engine”

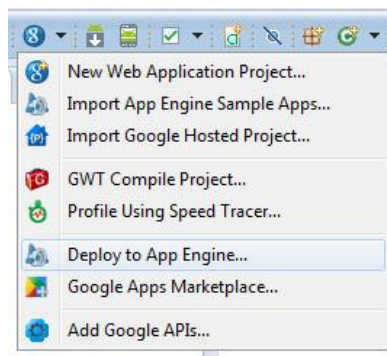


Ilustración 33 - Deploy hacia GAE

Una vez hecho click sobre “Deploy to App Engine”, se abrirá una nueva ventana en la cual seleccionar el proyecto a desplegar, importado desde la siguiente opción para Eclipse:

- File > Import
- Existing Projects to Workspace
- Seleccionamos el proyecto.
- Apply

Ahora únicamente, tendríamos que hacer click sobre el botón “Deploy to App Engine...” y esperar al despliegue de la aplicación.

12.2 Manual de instalación del cliente

Para la instalación de Android utilizaremos el entorno de desarrollo Eclipse, el cual permite instalar las herramientas necesarias para el desarrollo de la aplicación.

Seleccionaremos el Android SDK Manager, que aparecerá en la barra de herramientas para Eclipse:



Ilustración 34 - Manager SDK

Se desplegará Android SDK, donde tendremos que seleccionar los últimos componentes habilitados para el desarrollo SDK:

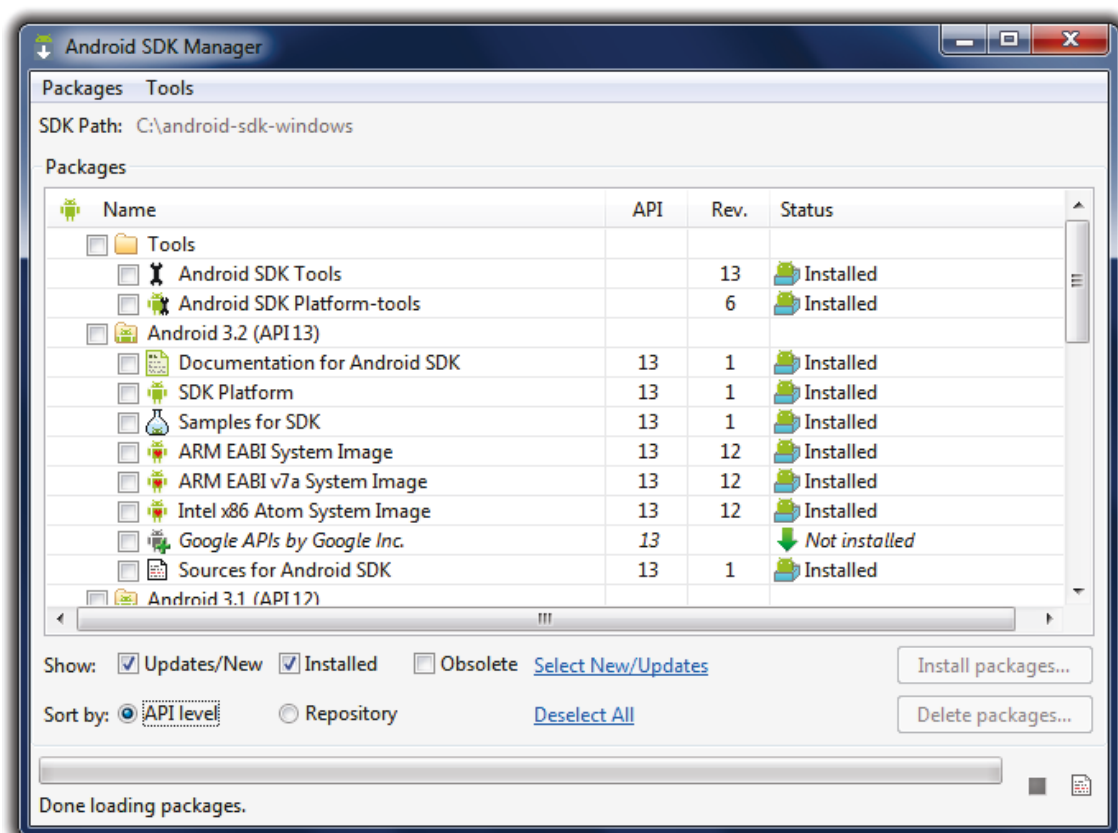


Ilustración 35 - Android SDK Manager

Una vez instaladas las herramientas, podremos importar el proyecto en Eclipse haciendo uso de:

- File > Import
- Android > Existing Android Code Into Workspace

12.3 Manual de uso del cliente

Al iniciar por primera vez la aplicación, nos encontraremos con esta pantalla

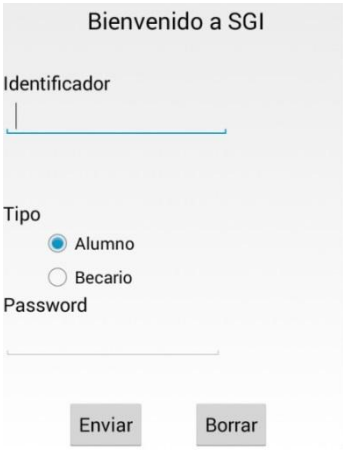


Ilustración 36 - Pantalla Inicio

En la que tendremos que introducir las siguientes credenciales para lograr acceso:

Tabla 51 - Roles

Nombre	Password	Rol
Usuario1	Contras1	Alumno
Master	Comander	Becario

En función del nombre y password de usuario introducimos, la aplicación nos re-direccionará a un determinado menú.



Esta imagen, corresponde al menú de becario – administración.

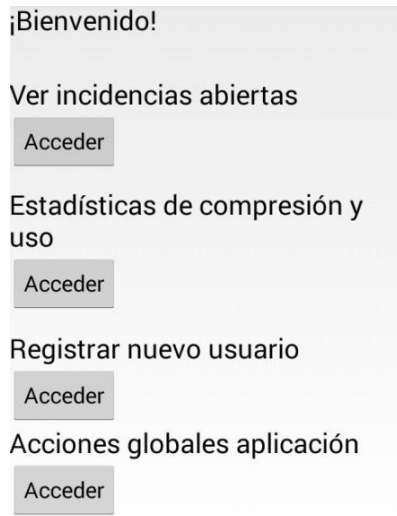


Ilustración 37 - Menu 1

O podemos acceder al menú de alumno.

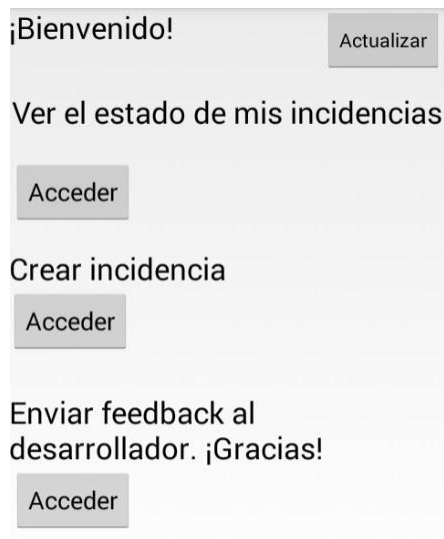


Ilustración 38 - Menu2



La generación de incidencias, se realiza desde la siguiente pantalla.

Generador Incidencias

Nombre de la Incidencia

Tipo

☒ Hardware

☐ Software

☐ Otros

Breve descripcion

Enviar Borrar

Ilustración 39 - Generar incidencia.

13. TRABAJOS FUTUROS

El trabajo fin de grado ha conseguido cumplir los objetivos buscados para la consecución del mismo, se ha logrado desarrollar una aplicación (cliente + servidor) con facilidades de uso y expansión hacia líneas futuras.

Paralelamente, y a título personal, ha servido para asentar los conocimientos de las estructuras dentro de aplicaciones móviles y PaaS, adquiriendo conocimientos nuevos aplicables.

El desarrollo de los subcomponentes de la aplicación, supone una acción recursiva (en cuanto al *feedback*) que permiten constantemente nuevas ideas sobre actuación para la mejora constante de la aplicación.

La compresión de datos, en la parte del servidor, representa una gran línea para trabajos futuros. Buscando diversos algoritmos de compresión que faciliten la fluidez en el tiempo, para el intercambio de datos.

Se comprende, una mejora constante de la interfaz, hacía líneas más fluidas e intuitivas para el usuario. Los tiempos permiten una mejora constante en las aplicaciones móviles, como hemos podido observar en el estudio inicial sobre el cambio de mercado para dispositivos móviles

Durante el desarrollo de los subcomponentes (cliente + servidor) para la aplicación en su totalidad, se han planteado las diversas líneas futuras que pueden implantarse a lo largo del tiempo:

- Mantener en constante actualización el servidor GAE y las versiones de desarrollo Eclipse – Android, que permitirán una evolución constante de la aplicación para mejor.
- Optimizar el servidor de forma directa, buscando algoritmos de mayor rendimiento. Para ello, se valorarán las alternativas sobre Google App Engine.
- Optimizar el lenguaje de programación constante, incluyendo código en otros lenguajes de programación como C++ o Python.
- Derivar los cálculos más intensivos para el manejo de la aplicación al servidor, añadiendo y optimizando las funcionalidades para el ahorro de datos, como una caché en el servidor.
- Optimización de las acciones de la aplicación y el servidor, que repercuten directamente en una optimización de la batería.

Implantando estas líneas futuras, podremos definitivamente implantarlo en el ciclo de vida sobre la Universidad, siendo empleada por el personal docente y alumnado. Al haber constituido una memoria documentada y un código documentado, la mejora de la aplicación puede ser constante.



14. AGRADECIMIENTOS

A las personas que me han rodeado y han conseguido proveerme de motivación para la consecución exitosa de este proyecto. Gracias a todos.

Empezando por mi familia hasta mi tutor, sin ellos y su apoyo, esto no hubiera sido posible.

Mi familia, por todos esos ánimos y fuerzas de incalculable valor, sin esperar nada a cambio.

Gracias a Rafa y Luismi, mis tutores, por enseñarme el camino de un ingeniero exitoso dentro de un proyecto exitoso.

Gracias a todas esas personas que han creído en mí, porque ellos me han dado fuerzas a superarme.

De nuevo, a todos ellos, gracias.



15. BIBLIOGRAFÍA

Bibliografía

1. **StackOverflow**. [En línea] <http://www.stackoverflow.com/questions/tagged/android>.
2. **Media, O'Reilly**. *Programming Google App Engine*.
3. **Soriano, Jose Enrique Amaro**. *Android. Programación de dispositivos móviles a través de ejemplos*.
4. **Gironés, Jesús Tomás**. *El Gran Libro de Android*.
5. **Soriano, José Enrique Amaro**. *El Gran Libro de programación Android avanzada*.
6. **Carmona, Antonio Morales**. *Programación Android paso a paso para principiantes*.
7. **Microsoft**. Microsoft Visual Studio. [En línea] <http://www.microsoft.com/visualstudio>.
8. **Google**. Google Developers. [En línea] <https://developers.google.com/appengine/>.
9. —. The Android Source Code, Governance Philosophy. [En línea] <http://source.android.com/source/index.html>.
10. **Microsoft**. Office Manual. [En línea] <http://office.microsoft.com/es-hn/word-help>.